Query-Driven Adaptive Sampling

by

Benjamin James Ayton

B.S. Aerospace Engineering, University of Texas at Austin (2015)
B.S. Physics, University of Texas at Austin (2015)
M.S. Aeronautics and Astronautics, Massachusetts Institute of Technology (2017)

Submitted to the Department of Aeronautics and Astronautics and the Joint Program in Applied Ocean Science & Engineering
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
and the
WOODS HOLE OCEANOGRAPHIC INSTITUTION
September 2022

Author .....
Department of Aeronautics and Astronautics, MIT
Applied Ocean Science & Engineering, WHOI
August 12, 2022

Certified by .....
Brian Williams
Professor of Aeronautics and Astronautics, MIT
Thesis Supervisor

Certified by .....
Richard Camilli
Associate Scientist with Tenure, Department of Applied Ocean Physics & Engineering, WHOI
Thesis Supervisor

Accepted by .....
Jonathan P. How
R.C. Maclaurin Professor of Aeronautics and Astronautics, MIT
Chair, Graduate Program Committee

Accepted by .....
David Ralston
Associate Scientist with Tenure, WHOI
Chair, Joint Committee for Applied Ocean Science & Engineering

Query-Driven Adaptive Sampling

by

Benjamin James Ayton

Submitted to the Department of Aeronautics and Astronautics, MIT
Applied Ocean Science & Engineering, WHOI
on August 12, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Automated information gathering allows exploration of environments where data is limited and gathering observations introduces risk, such as underwater and planetary exploration. Typically, exploration has been performed in service of a query, with a unique algorithm developed for each mission. Yet this approach does not allow scientists to respond to novel questions as they are raised. In this thesis, we develop a single approach for a broad range of adaptive sampling missions with risk and limited prior knowledge. To achieve this, we present contributions in planning adaptive missions in service of queries, and modeling multi-attribute environments.

First, we define a query language suitable for specifying diverse goals in adaptive sampling. The language fully encompasses objectives from previous adaptive sampling approaches, and significantly extends the possible range of objectives. We prove that queries expressible in this language are not biased in a way that avoids information. We then describe a Monte Carlo tree search approach to plan for all queries in our language, using sample based objective estimators embedded within tree search. This approach outperforms methods that maximize information about all variables in hydrocarbon seep search and fire escape scenarios. Next, we show how to plan when the policy must bound risk as a function of reward. By solving approximating problems, we guarantee risk bounds on policies with large numbers of actions and continuous observations, ensuring that risks are only taken when justified by reward.

Exploration is limited by the quality of the environment model, so we introduce Gaussian process models with directed acyclic structure to improve model accuracy under limited data. The addition of interpretable structure allows qualitative expert knowledge of the environment to be encoded through structure and parameter constraints. Since expert knowledge may be incomplete, we introduce efficient structure learning over structural models using A* search with bounding conflicts. By placing bounds on likelihood of substructures, we limit the number of structures that are trained, significantly accelerating search. Experiments modeling geographic data show that our model produces more accurate predictions than existing Gaussian process methods, and using bounds allows structure to be learned in 50% of the time.

Thesis Supervisor: Brian Williams
Title: Professor of Aeronautics and Astronautics, MIT

Thesis Supervisor: Richard Camilli
Title: Associate Scientist with Tenure, Department of Applied Ocean Physics & Engineering, WHOI

Acknowledgments

First, I'd like to thank my wife, Sylvia Dai, from the bottom of my heart. Her care and support is the only reason I got through this experience in one piece. Whenever I needed to vent, or was confused, she was always there to listen. Her tireless ambition and unwillingness to accept failure was exactly the push I needed. Plus, Sylvia taught me more about writing a good technical paper than perhaps anyone else.

I'd like to thank my advisor Brian Williams. Brian gave me a chance, despite knowing little of my background, and forged for me an opportunity to work on a project that matched my interest. I'll always be grateful for him giving me a place at MIT, and allowing me to explore my passions.

I'd like to thank my advisor Rich Camilli. Rich is one of the most storied and simply impressive people I have ever met. I hope to one day live a life that is one tenth as interesting as his. In addition to being the organizer of all those glamorous trips, Rich always ensured that my work had a real purpose, and made sure I was well grounded in reality. He always made it clear that someone was there who cared, and I appreciate the effort he made to make sure my next step in life was what I wanted. Sorry I'm not going into academia or starting a company, Rich. Maybe one day.

I'd like to thank my thesis and defense committee members, including Youssef Marzouk for his insight, and Yogi Girdhar for stepping up and playing the role of my defense chair. I'm extremely thankful for his kindness. I'd like to thank Masataro Asai for being a wonderful mentor during an internship, and a thesis reader. Masataro is the hardest working person I've ever met, and always contributed far more than his fair share for the projects we worked on. I hope I'll have the chance to work with you again Masataro. I'd like to thank Claudio Toledo for being my final thesis reader. Claudio always seems to be in a good mood, and his excitement is infectious.

In addition to committee members, there were so many people that were instrumental to this PhD. Thank you to all Mersians that shared my tenure, new and old; Amy Phung, Allen Wang, Andrew Wang, Cameron Pittman, Charles Dawson, Cyrus Huang, Dan Strawser, Delia Stephens, Enrique Fernandez Gonzalez, Eric Timmons,

Jacob Broida, Jingkai Chen, Marlyse Reeves, Matt Deyo, Matthew Orton, Meng Feng, Nan Ma, Nick Pascucci, Nikhil Bhargava, Pedro Santana, Peng Yu, Sang Lee, Simon Fang, Steve Levine, Sungkweon Hong, Tesla Wells, Viraj Parimi, Weiqiao Han, Yuening Zhang, Yui Sujichantararat, and Zach Duguid. They were always there to share the good times and to complain with me during the bad. The amazing things they do with their lives are endlessly inspirational. I'd also like to thank Christian Muise and Tiago Vaquero for their leadership and mentorship, particularly in the early days of my PhD. A special thanks goes to members of the Scouts team for their Herculean efforts in making deployment of our technology a reality, and the Spock team for their interest and passion in carrying forward this work.

Many scientists, collaborators, and sponsors also channeled their own energies and enthusiasm into this project, making it more than it could have been otherwise. I'd like to thank them for making my work better, and pushing it into new directions. These include champions from ExxonMobil, including Lori Summa and Michael Braun, who helped inspire the direction of query-driven adaptive sampling. Thank you also to my collaborators from Vulcan; Louisa van Zeeland, Sam McKennoch, and Scott Smith, for their excitement, and helping me apply Spock to problems with a positive benefit on the world.

Thank you to collaborators and cruise-mates, who all helped me to build technology with real utility. My geosciences collaborators Lori Summa (again), Paraskevi Nomikou, and Peter Vrolijk always believed in my work, put aside their own scientific goals to test my technology, and published alongside me. When life is a little less crazy, I'd like to really be able to contribute to your additional projects and ideas. My robotics collaborators Angelos Mallios, Gideon Billings, Jackson Shields, Mike Jakuba, Oscar Pizarro, Stefan Williams were critical to masking field deployments a success, and provided unique insights into how the work in this thesis could be further improved.

I have fond memories with each and every person listed above, and there is much more that I could say. Thank you for joining me on this journey, and I look forward to what comes next.

The work in this thesis was supported by the Exxon Mobil Corporation as part of the MIT Energy Initiative under the project 'Autonomous System for Deep Sea Hydrocarbon Detection and Monitoring', NASA's PSTAR program under the project 'Cooperative Exploration with Under-actuated Autonomous Vehicles in Hazardous Environments', and the Vulcan Machine Learning Center for Impact under the project 'Machine Learning Based Persistent Autonomous Underwater Scientific Studies'.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Thesis Overview

Robotic and autonomous systems allow humanity to observe parts of the world that they could not otherwise explore, like the depths of the ocean and surfaces of other planetary bodies. Reaching these remote areas is expensive, technically challenging, and risky, so there is a need for a decision making capability that performs exploration efficiently and safely. In order to do so, we claim in this thesis that autonomous exploration must be able to account for the following challenges:

1. **Exploration in service of a query.** A mission is typically conducted in service of a specific query, such as something to find or a question to be answered. The query influences which observations are most useful. An autonomous exploration system should be aware of the query and evaluate the utility of observations in the context of the query to be solved.

2. **Queries prompted by new observations.** The journey for scientific understanding rarely ends with a single answer. Far more frequently, an answer raises many more questions worthy of further investigation, which are not considered until observations are received. It is impossible to predict queries in advance, so an autonomous exploration system must be flexible in query input.

3. **Risk during exploration, with complex interactions between risk and reward.** Gathering observations frequently leads to unavoidable risk of unacceptable damage to an observing agent or to the environment it is exploring.

The risk of those outcomes occurring must be controlled, and traded off against the fact that stricter guarantees on safety may result in less useful information. It is often not clear, however, how changing acceptable risk will affect information that can be gathered.

4. **Limited data availability.** A more accurate model of the environment being explored reduces uncertainty in answers to queries. Unfortunately, adaptive exploration is normally performed in environments where very limited data is available, so there is significant noise when learning models and making predictions from data.

This thesis proposes an approach to autonomous adaptive exploration called 'Spock' that addresses these challenges. Spock is query-driven, in that it allows users to specify their exploration goals in terms of what they wish to learn or answer, and tailors exploration towards that query. This thesis develops a query language for Spock that allows for a great deal of flexibility in goals, so that the query does not need to be known in advance. Spock can reason about the risk that will be encountered by an exploring agent, and allows users to specify tolerable risk as a function of the information that will be gathered. Spock uses an environment model that can capture expert knowledge of interactions between variables in an environment, in order to improve the quality of inference when data is limited. Spock can also be used to learn parts of the environment model that are unknown, but only introduces variable correlations that are strongly justified, further improving prediction quality.

1.1 Need and Motivating Scenario

In this section, we further elaborate on each of the previously mentioned challenges, and motivate them in the context of a scenario where an underwater vehicle is tasked with predicting the number of hydrocarbon seeps in part of the environment. This is far from the only application of Spock; in the development of this thesis Spock was successfully deployed on a number of field campaigns and examined in multiple case

studies. We provide further examples of applications of Spock in Section 1.6.

In certain areas of the ocean, subsurface hydrocarbon deposits naturally leak from openings in the sea floor, leading to hydrocarbon seepage [24]. Hydrocarbon seeps support unique fauna [124], and give information about the subsurface geology in the local area [95]. As a result, locating seeps and determining whether they are active is of interest within biology and geology, in addition to group interested in natural resource extraction.

We consider an autonomous underwater vehicle that is tasked with exploring a region of the ocean, and must take observations that best inform a prediction of how many seeps exist at depths below 500 m. The mission's measure of success is the probability of the mode of how many seeps deeper than 500 m exist after all observations have been taken. For example, observations that result in a posterior belief that there are 5 seeps with 20% probability and 6 seeps with 80% probability are preferred to observations that result in a posterior belief that there are 4 seeps with 10% probability, 5 seeps with 40% probability, and 6 seeps with 60% probability.

Seeps can be detected using visual and spectrometer confirmation of hydrocarbon release. While confirmation of seepage can only be performed at close range [135], seeps are commonly associated with sea floor features that can be detected by long range sonar [114]. These features can be used to select locations to focus exploration. The geological processes that result in seeps result in the formation of mounds, pockmarks, and ridges [69]. Additionally, microbial metabolism on hydrocarbons forms carbonates, which reflect more sound than surrounding rock and are identifiable from sonar. All these features are expected to be correlated with seepage, but the quantitative strength of those correlations is unknown. In order to determine how many seeps are present in the environment, the autonomous vehicle must visit sites with different combinations of bathymetric features and measure whether seeps are present, use that information to determine how feature presence is related to the presence of seepage, and use those learned relationships to estimate the number of active seeps in the environment. A visualization of this scenario is shown in Figure 1-1, where different combinations of bathymetric features indicate potential seep sites

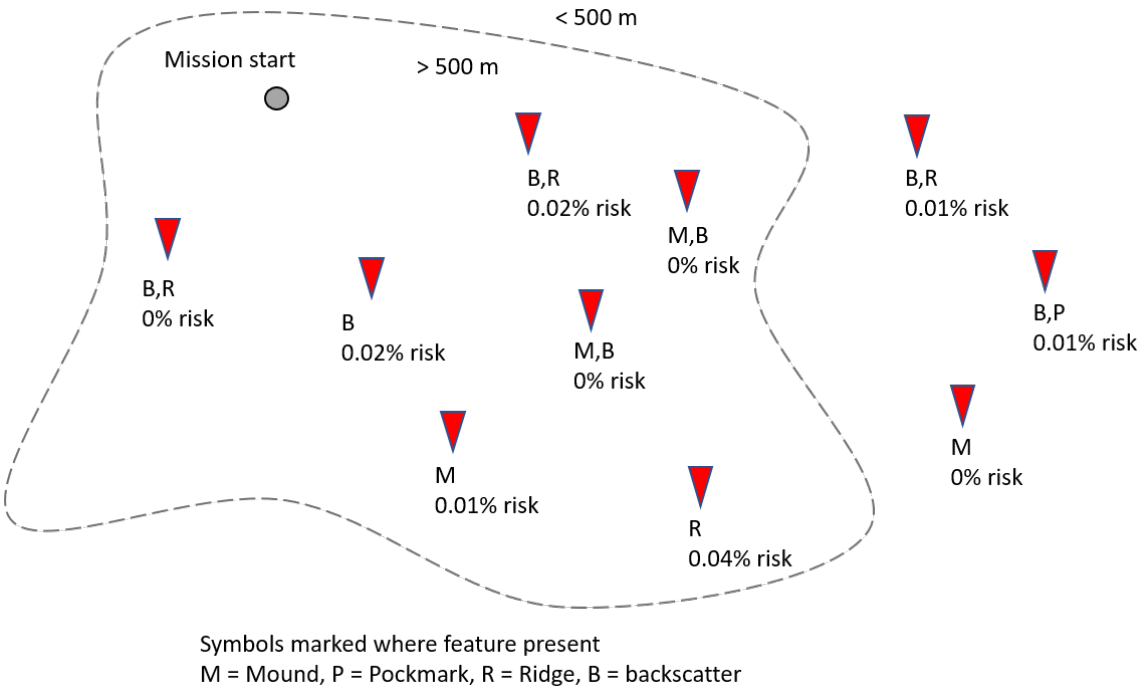


Figure 1-1: Spock motivating scenario, where locations must be observed to maximize the mode of the number of seeps within the 500 m depth contour. Red arrows indicate candidate seep sites, and letters indicate local features that may be correlated with seeps.

to be explored. In this figure, the objective is to maximize the posterior mode of the number of seeps inside the 500 m depth contour.

We now describe how the scenario leads to each of the four challenges we identified.

The quality of observations is determined by the query. An observation is always more useful when it results in a more precise answer to the question that motivated the study. In this case, the observation is most useful when it increases belief in the most likely number of seeps deeper than 500 m. Assuming that the presence or absence of bathymetric features throughout the environment is already known and only the presence of seeps is uncertain, it will be most informative to visit sites with bathymetric features that are common below 500 m, and those with large uncertainty in how they relate to the presence of seepage. Determination of the relationship between features that are common below 500 m and seeps will improve predictions of seepage at more locations, and measuring features with uncertain relationships to

seepage will better improve the quality of predictions.

This is a very different strategy compared to maximizing information about the existence of seeps across the whole environment. In this query, we don't care about improving estimates of seepage at locations above 500 m. Features that only appear above 500 m can effectively be ignored. This behavior is not coded as constraints that are input to Spock. Instead, ignoring features that only appear above 500 m is inferred from the query and simulation of how observations could answer it.

Queries may arise as the mission progresses and cannot be predicted. In theory, an algorithm could be developed specifically to estimate the number of seeps below 500 m, and then deployed. But over the course of exploration, new and unexpected observations occur, which prompts new questions. The underwater vehicle may capture an image of an unfamiliar species, and biologists that receive that information may want to understand the types of bathymetric features that make up that species' habitat. This query could not be anticipated in advance, but it would be highly beneficial to be able to produce a new plan to answer it, since an observing agent is already in the area. To allow this capability, we propose that adaptive sampling should produce plans directly in response to queries input by the user, and allow a great deal of flexibility in their expression. This way, scientific questions prompted by exploration can be answered as they arise, without needing to generate new planning approaches in response to every question.

Bounds must be placed on risk, which are informed by information gathered. The autonomous vehicle in the motivating scenario may be subject to unpredictable currents as it navigates, and this introduces uncertainty into its position estimation. Attempting to navigate close to seeps that are located in tight valleys, or next to extreme slopes, exposes the vehicle to a risk of collision with the ocean floor, which could cause loss of the vehicle. However, this risk is unavoidable, because the only way to avoid all risk is to stay so far away from possible seep sites that the presence of seepage could not be confirmed or denied. The degree of acceptable risk

must therefore be bounded.

However, the level of acceptable risk is often determined by the efficacy of the mission. Intuitively, risks should be taken only if they are 'worth it'. A moderate risk of 0.01% may be acceptable if the most likely number of seeps below 500 m could be determined with more than 80% certainty. But that level of risk may be unacceptable if the most likely number of seeps could only be determined with 40% or less. Reasoning about the relative utility and risk of observations in this way allows a mission to be performed that is both practical and informative, and takes risks when necessary and avoids them otherwise.

Data is limited, but utilizing expert knowledge can improve prediction. A more accurate model enables predictions of future observations and unseen locations to better match reality. In order to predict the number of seeps below 500 m accurately, it is necessary to have accurate predictions of the presence of seepage at locations that were not observed. Due to the expense of exploration, previous observations may have only been taken at 100 locations or fewer, which makes prediction difficult. In particular, statistical noise will cause correlations to be observed between features and seeps, even if they are independent. Since data is limited, these statistical correlations can be significant, and can reduce prediction accuracy.

However, geologists may already have scientifically-motivated evidence that certain features are correlated with seepage. If the environment model is able to capture this prior knowledge as a constraint, then predictions can be enforced to be consistent with known science and be more accurate. To do this in practice, the environment model must be interpretable, so that the constraints between variables can be directly encoded.

1.2 Problem Statement

Given the example scenarios that motivated this work and the challenges we seek to address, we now provide a high level overview of the problem solved by this thesis.

More specific details on each aspect of this problem are provided in the technical chapters.

The problem solved by Spock consists of four key elements; a model of an agent that performs exploration, including its sensors and dynamics, an environment that must be modeled while satisfying user specified constraints, a query about the environment that must be answered, and a bound on the risk that the agent can take on its mission. The problem is solved by executing a plan that takes observations of the environment to best answer the query, while constraining risk to be below the bound.

We consider a mobile agent, like an underwater vehicle, that navigates an environment with multiple attributes at each location. For example, seep presence and mound presence that attributes at each location in the seep search example. Each single attribute is spatially and temporally correlated between different locations, so that a seep could be more likely at a location that is close to another known seep. Different attributes are correlated with each other, so that a seep at a location may be more likely if elevated backscatter is also present at that location. These combined correlations allow prediction of the environment at locations that have not been observed.

Prior to starting a mission, a limited amount of prior observations of the environment are available, such as where seeps, mounds, or backscatter were previously known to exist. In addition, domain experts provide qualitative constraints on known relationships between variables, such as seeps being causally related to mounds, and the relationship between the two being monotonic, so that as the probability of a mound increases, the probability of seepage increases. The constraints and data are used to construct a stochastic model for the environment at unobserved locations.

The agent may take a number of different actions that change its position. After taking an action, the agent's position is updated according to a model of its dynamics, like moving it to a new candidate seep site. Taking an action may also incur some risk of a failure condition occurring, such as collision with the environment or becoming stuck. Once the agent reaches a new location, it then takes an observation of the environment at that location. An observation could be whether a seep is present, or

a noisy function of that signal, for example. The observations are used to update the model of the environment and to give a better answer to a query.

In addition to an agent and a description of the environment, our problem considers a query that should be answered about the environment. The query in our motivating scenario was to optimize the posterior mode of belief over the number of seeps below 500 m. The form of the query is flexible to allow for a new objectives prompted by observations, like determining the habitat of a newly discovered species.

Finally, we also consider a risk bound on acceptable probability of failure, expressed as a function of overall achievement or precision of the answer to the query, for example, accepting more risk for higher modes of the number of seeps below 500 m. We seek execution of an adaptive policy that selects actions and observations to be performed by the agent, which may change in response to the previous observations gathered. The policy should maximize the objective specified by the query, while bounding the total risk of the policy to be below the risk bound applied to the objective value.

1.3 Spock in Action

In this section, we describe how the seep exploration problem described above could be set up in Spock, and some of the behavior that it would exhibit in a hypothetical scenario.

In order to formulate this problem in Spock, a user begins by describing what variables exist in the world, and how they are known to be related. The user encodes that the environment contains seeps, mounds, pockmarks, ridges, and elevated backscatter, all modeled as binary random variables at different locations in $(lon, lat, depth)$ space. Geologists state that backscatter and mounds are known to be positively correlated with seepage, and while the effect of pockmarks and ridges is uncertain. Constraints are also provided that if correlations do exist, seepage causally depends on the bathymetric features. The observed presence of seepage at locations that were visited is also provided as an input to help train the model.

The user then provides Spock with a query that details the objective to maximize the posterior mode of the number of seeps below 500 m. Input also describes the dynamics of the vehicle, and how its position uncertainty grows with time. Finally, the user describes that they would accept up to 0.05% risk if executing the plan would perfectly determine the number of seeps below 500 m, and only 0.001% risk if the mode of the number of seeps below 500 m is effectively 0.

This method of operations is in stark contrast to the setup and planning that has been used in prior adaptive sampling deployments. In those deployments, the query and environment model are both hard-coded prior to a mission, and the strategy of where to measure is strongly coupled to both. If a new bathymetric feature was identified to be an important indicator of seepage, it would not be straightforward to add it to the environment model, and there would be no guarantee that the decision making algorithm would be able to account for it correctly. As understanding of the environment evolves, additional expert knowledge and new queries would not be able to be included or considered on the deployment. Finally, risk would not be accounted for explicitly, and mission operators would need to establish forbidden regions for the vehicle. These ultimately may still allow a mission that does not satisfy the risk bounds, or may result in a large degree of conservatism.

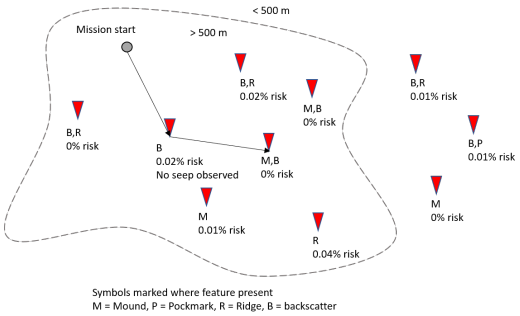
Spock takes the known environment variables, statements of expert knowledge, and a query, and it begins to construct internal models of how variables in the environment are related. It has been told that mounds and backscatter are positively correlated with seepage from expert constraints. In the prior data where the presence or absence of seeps were observed that is available to Spock, very few sites had pockmarks and most had mounds. Without observations of pockmarks or a lack of mounds, whether or not pockmarks and mounds have any effect on seepage is highly uncertain. On the other hand, whether a ridge was present is observed to be effectively independent of seepage on many observations, and so the belief that seeps are not correlated with ridges is strong.

Spock proposes candidate models where seeps are and are not causally related to pockmarks and ridges, and probabilities for those models. Spock reasons that, based

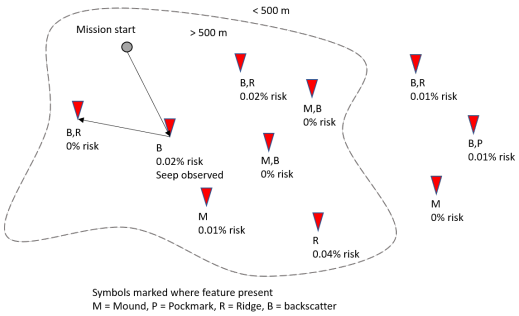
on the data, the models where ridges are correlated with seeps have low probability, while whether or not pockmarks are correlated with seeps has relatively equal probability. Mounds and backscatter are known to be correlated with seepage by expert knowledge, but the quantitative effect of mounds is highly uncertain. Let us say that in the prior data, seeps always occurred when elevated backscatter was present, and rarely occurred when it was not. Spock therefore infers a strong positive effect of backscatter on the presence of seepage.

Spock uses those models to produce simulations of where seeps exist in the environment, in order to learn what would be observed if an agent were to visit each of the locations, and how those observations would affect belief in the number of seeps below 500 m. From those results, Spock is able to determine how each observation would contribute to answering the query. All the candidate locations for seeps below 500 m do not have pockmarks, so even though there is significant uncertainty in whether the presence of a pockmark would significantly affect seep predictions, the simulations reveal that it doesn't significantly change the number of seeps below 500 m. On the other hand, several locations below 500 m do not have mounds, so quantifying the impact of mounds significantly improves belief in the most likely number of seeps below 500 m.

Based on these observations, Spock selects to first observe a location with no mound and positive backscatter. This combination is chosen because if a seep is not present, it would be strong evidence that the lack of mounds is an indicator for no seepage, and belief in the number of seeps below 500 m would be refined significantly. There are three such sites, one that can be observed without risk that is far away from other sites, and two that can be observed with 0.02% risk that are more central. The risky sites are preferred, because they allow the agent to reach sites with stronger spatial correlations on following observation. The risk of the observation is compared against the impact on predictions of seeps that could occur, and it is determined that the tradeoff between risk and belief update is acceptable under the risk bounds. An observation is taken, and Spock uses that information to update its models and predictions for the number of seeps below 500 m.



(a) Mission executed after observing no seep at first site.



(b) Mission executed after observing no seep at first site.

Figure 1-2: Example missions executed by Spock in the motivating seep search scenario. When no seep is found at the site of first measurement with backscatter and no mound, the effect of mounds on seepage is inferred to be strong. When a seep is found at that site, finding a location with backscatter, no mound, and no seep would still significantly change belief over the number of seeps below 500 m, but searching for one is no longer worth 0.02% risk.

Spock's next action then depends on what was observed after the first action. In the first case, no seep is present, as in Figure 1-2a. This is a significant result, since seeps were always observed when backscatter was present in prior data. It's likely that this means that the correlation between mounds and seeps is also strong quantitatively, and that seeps should be predicted to be less likely when mounds are not present. Since the uncertainty in the impact of mounds has been determined, simulations reveal that the next most informative measurement would be to measure a location that is centrally located. Presence of seepage is expected to be spatially correlated, and this location would best refine the estimate for the number of seeps.

In the second case, a seep is present, as in Figure 1-2b. There now remains some uncertainty about whether no mounds has a comparable effect on seep presence to backscatter and the observation was unlucky, or whether the impact of no mounds on seeps is very weak. Observing no seep at another site with backscatter and no mounds would still refine belief over the number of seeps below 500 m, but based on the last observation, that outcome is not likely. Spock's simulations reveal that the lower likelihood of no seep means that taking an additional 0.02% risk is no longer justified, and the best action is chosen to be the site with no mound and backscatter

with no risk.

Regardless of the actions taken, after the vehicle takes a set number of observations, the mission ends. The vehicle surfaces, and reports back belief over the number of seeps below 500 m based on its observations.

This scenario demonstrates Spock determining the quality of observations based on relevance to the query, by choosing not to determine the effect of positive presence of a pockmark on seepage, and to focus on locations without mounds. Risk is constrained as reward, and risky seep locations are only observed when they are likely to provide meaningful information about the presence of seeps elsewhere below 500 m. The choice of locations is guided by expert knowledge, in this case about the positive correlations between backscatter, mounds, and seepage. Finally, a similar mission could be performed with a variety of different queries as input, prompted by discoveries as the mission progresses.

1.4 An Overview of Spock

We now provide a summary of a Spock as a system, and provide some more detail into how it operates. Spock is constructed from a number of components, and those components will be developed in the technical chapters of this thesis.

At a high level, Spock consists of a planning component and an environment modeling component, as shown in Figure 1-3. The planning component makes decisions on where and what an agent should observe, and uses the environment modeling component to inform those decisions. The environment modeling component generates models that are consistent with expert knowledge, and allows predictions of the value of unobserved environment variables.

The planning component takes as input a query that follows a query language described in Chapter 2, and a risk bound expressed as a function of reward. The query defines how reward should be computed for the plans produced by Spock. The planning component is responsible for producing an online plan that optimizes an objective derived from the query, while ensuring that the risk taken across all executions

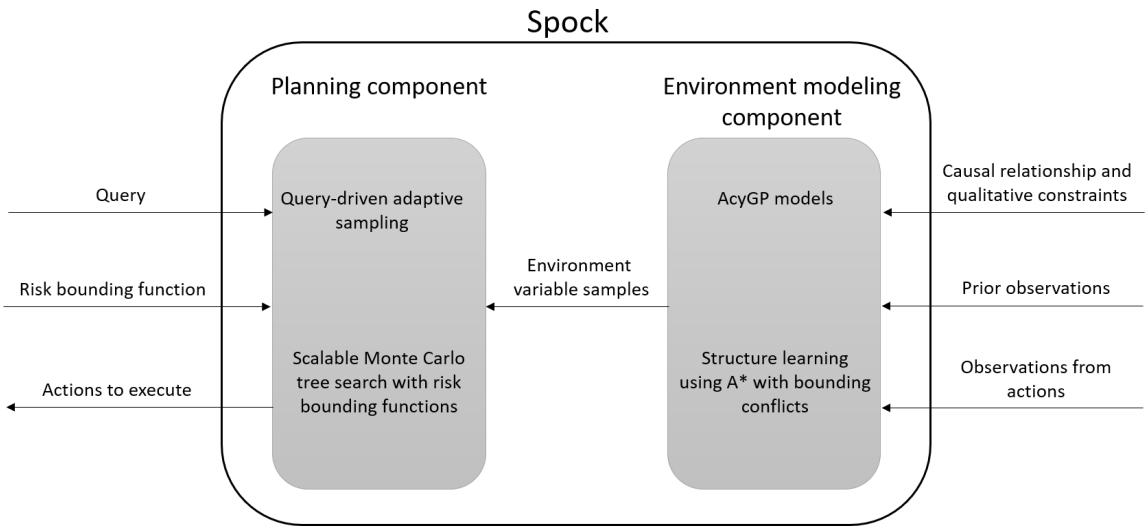


Figure 1-3: Overview of Spock as a system. The planning component chooses actions to execute for a given query and risk bound. The environment modeling component uses expert knowledge and observations to build a predictive model of the environment, and provides samples of that model to the planning component.

of the plan is bounded by the risk bound applied to the expected reward. Planning is performed using Monte Carlo tree search in an online manner, by constructing a series of unconditional plans.

The planning component makes use of two technologies: query-driven adaptive sampling and scalable Monte Carlo tree search with risk bounding functions. Query-driven adaptive sampling is described in Chapter 3, and details how reward can be computed for general queries using samples drawn from the environment model. General reward estimators are embedded in the Monte Carlo search tree, so that planning can be performed with a single technique for many queries. Scalable Monte Carlo tree search with risk bounding functions is discussed in Chapter 4, where it is shown that the online plans can be guaranteed to satisfy risk bounds expressed as functions by placing constraints on actions in the online plans. The constraints are formulated so that risky actions can be taken if they are likely to yield future reward.

The environment modeling component takes as input user-defined constraints on the presence or absence of causal relationships between different environment attributes, qualitative constraints on those relationships, such as monotonicity, and observations that were collected in past or during the mission. The environment

modeling component then produces samples of the environment that can be used within the planning component.

The environment modeling component also uses two related technologies: AcyGP models and structure learning using A* with bounding conflicts. The AcyGP model uses multiple Gaussian processes connected through a directed acyclic graph structure, and is introduced in Chapter 5. The AcyGP model makes it simple to express qualitative relationships and constraints on structure. The remainder of the structure that was not constrained by users is optimized through structure learning using A* with bounding conflicts, as described in Chapter 6. A* with bounding conflicts uses bounds to allow the optimal structure to be found without evaluating all possible structural models, and describes when models should be evaluated in the course of search.

1.5 Summary of Contributions

We now provide a summary of the major contributions provided by this thesis, and describe how they help to achieve our overall problem statement.

A Broad Query Language for Adaptive Sampling The literature on adaptive planning currently describes a large number of approaches to adaptive sampling problems. These algorithms are specific to certain objectives and models, and are not easily generalized. With query-driven adaptive sampling, our goal is to be able to provide a single approach to adaptive sampling in service of many possible queries provided by the user. A key step for providing this capability is defining the space of queries that are suitable for adaptive sampling, and the means by which they may be input to the algorithm.

In Chapter 2, we construct queries from a combination of one of three choices of objective, a user defined function, and an optional sufficient condition. The function computes a variable of interest from the environment model, and the objective of the mission is to maximize the objective applied to the variable of interest, or

if the sufficient condition is supplied, minimize the number of observations required for the objective to reach the sufficient condition. We then provide modifiers called 'specializations' that further affect how the objective is applied. Overall, the query language encompasses most known adaptive sampling objectives, while introducing significantly more capability of expression over the state of the art. We then demonstrate that it is possible to formulate queries with optimal solutions that deliberately ignore data that can be obtained for free, but we prove that all queries specified in our query language do not lead to this behavior. The introduction of queries provides a framework for specifying a broad range of adaptive sampling goals, while ensuring they are well-posed.

Monte Carlo Tree Search for Complex Queries We adopt a Monte Carlo tree search (MCTS) approach to solve for policies for what observations to take at what time. Unlike most existing adaptive sampling algorithms, the flexibility in our objectives means that they cannot necessarily be computed in closed form, and instead must be estimated using computationally intensive estimators. Our approach must balance the time taken exploring the space of possible actions against the time required to estimate the result of the query after any given sequence of actions. The introduction of sufficient conditions and specializations further complicates search, as missions can end as soon as the sufficient condition is met, and specializations mean that certain outcomes can be ignored.

In Chapter 3, we introduce extensions to MCTS that account for the additional complexity introduced by a query-directed framework. To plan with computationally intensive queries, we embed sample based estimators of probability density and mutual information within an MCTS search tree. Action selection rules within MCTS are used to determine whether a sequence of actions is possibly optimal, so that more time is committed to making an objective estimate more accurate using additional samples. Meanwhile, coarse objective estimates that are likely to be low are not refined with additional samples. To allow planning with sufficient conditions, we introduce a mechanism that allows rollouts to terminate before reaching a leaf

state within MCTS. Finally, we also show how to compute which outcomes must be considered for the purpose of objective computation with specializations. Our Monte Carlo tree search method allows us produce a single algorithm that will work for all queries, and allows planning in an anytime manner.

Long Duration Risk Bounded Planning with Risk-Bounding Functions A risk-bounding function specifies an acceptable level of risk to the exploring agent as a function of the value of the observations that will be taken, so that the agent can trade off risk against reward. Guaranteeing satisfaction of the risk bound in a policy typically requires consideration of every possible outcome that could occur. For many practical problems with a large number of possible outcomes of any observation, the state space of search becomes large very quickly, which effectively limits the duration of policies to a handful of actions. Furthermore, we cannot consider every possible outcome of a continuous observation.

In Chapter 4, we show that it is possible to guarantee satisfaction of a risk bounding function for policies that depend on a continuous set of outcomes, and for missions with 20 actions or more. This is achieved by solving a sequence of smaller approximating problems that consider the average outcome of a sequence of actions, then replanning in response to each observation. The advantage of the approximating problems is that the generated search tree has a smaller, finite number of branches, even when there is a continuous space of outcomes of an action. This allows significantly larger problems to be solved. By placing carefully formulated constraints on those approximating problems, we show that a constrained solution to the next approximating problem necessarily exists, and that executing the first action of each approximating problem in sequence will necessarily lead to satisfaction of the risk bounding function. Risk bounded planning ensures that risks are only taken when they also result in worthwhile reward, and our contribution is to allow risk and reward tradeoff to be considered over long missions.

Gaussian Process Models with Directed Acyclic Structure Since adaptive sampling is used to reduce uncertainty about an environment or answer questions that remain unanswered, it is typically used in applications with significant uncertainty in the environment and few observations. Gaussian process models are widely used in adaptive sampling because they provide uncertainty estimates with their predictions. Precisely answering a query requires an accurate model of the environment, so it is advantageous to encode any information known about the environment outside of the data. Even if they cannot quantitatively describe relationships between variables, experts frequently possess qualitative knowledge of the form of variables that are modeled and the interactions between them, such as monotonicity relationships, causality, and independence. Predictive accuracy can be improved by ensuring that the environment model captures those relationships, but existing Gaussian process methods do not provide an interpretable way to enforce them.

In Chapter 5 we introduce a novel Gaussian process model with directed acyclic graph structure connecting the different attributes being modeled. This allows a user to model conditional independence between attributes and known causal relationships in an intuitive manner. The model further allows heterogeneous attributes, such as discrete or continuous variables, to be captured appropriately. Then, by placing constraints on parameters that connect attributes in the model, we are able to enforce qualitative constraints like monotonicity. Our Gaussian process model enables accurate environment modeling and prediction, even in the presence of very limited data.

Efficient Structure Learning Using A* with Bounding Conflicts Although the capability to capture qualitative constraints in an environment model is valuable, it is common that certain relationships may not be known by domain experts. In this case, it is appropriate to determine the most likely relationships and their probabilities using the observational data that is available. In the context of a model with graphical structure, search over the space of models is achieved through structure learning, which is well understood and operates by evaluating the likelihood of the data under

different structures. However, unlike most applications of structure learning, it is a highly time intensive process to determine the likelihood of a Gaussian process model under each structure, as the Gaussian process must be trained.

In Chapter 6 we show that optimal structure learning can be performed without evaluating the likelihood of all possible structures by using A* with bounding conflicts. When searching over the space of structures, a bound is placed on the likelihood by using the likelihood of any evaluated structure with more edges. In this way, certain structures can be proven to be outside of the set of k best without ever evaluating them. This greatly reduces the number of structures that need to be trained, and accelerates search for the most likely structure. Using A* with bounding conflicts this way speeds up model learning considerably, without needing to resort to approximation methods.

1.6 Applications of Spock

Before introducing Spock technically, we list some field deployments that made use of Spock, and case studies that were performed to examine possible future use cases. These scenarios show the diversity of applications that query-driven adaptive sampling easily supports, without algorithm modification, and shows that Spock has been successfully deployed in practice.

Coral Reef Species Monitoring Coral reefs are havens for unique species of coral and fish that are only found in those biomes [85, 109]. In order to ensure the preservation of those species, it is of interest to determine the extent and health of a coral reef, determining which species are present and whether they have been subject to harmful bleaching. Large coral reefs can cover many square kilometers of the ocean floor, so to conduct an extensive study without performing a large number of human dives, persistent autonomous underwater vehicles with optical and acoustic imaging can be used to map the reef over large areas [149, 150]. Since coral reefs are fragile, it is important for any exploring vehicle to limit the probability of colliding with the

reef as it travels over it due to unexpected currents or tall coral that was not detected by on-board sonar.

This application, in such a fragile ecosystem, demonstrated the critical role of careful consideration of risk during adaptive sampling. Technology described in this thesis was utilized for coral monitoring on a cruise to the ‘Au‘au Channel and the Big Island of Hawai‘i in January 2018 as part of NASA’s PSTAR project, with ship time provided by the Schmidt Ocean Institute [108]. Coral exploration missions performed by autonomous underwater vehicles were planned with an objective to maximize mutual information between coral density and photo observations, as well as the presence of certain coral species and observations. The probability of collision with the reef was estimated using position uncertainty derived from currents, and plans were developed to ensure that the risk did not exceed a user specified bound.

Characterization of Natural Hydrocarbon Seeps As described by the example scenario earlier in this chapter, characterization of natural hydrocarbon seeps required adaptive sampling to be performed in a domain with extremely limited prior knowledge, but incorporating expert knowledge given by geologists. Adaptive sampling and environment modeling approaches described in this thesis were used to search for hydrocarbon seeps in the Costa Rica active margin in December 2018 [144], supported by the Schmidt Ocean Institute, NASA PSTAR, and Navistry Corp. A structured model was constructed to capture geologists qualitative understanding of the relationships between bathymetric features, acoustic backscatter signals, water column acoustic anomalies, and the presence of seepage. The quantitative strengths of these signals was then learned from data under the asserted qualitative model. Adaptive planning was then used to produce dives with remotely operated and autonomous vehicles to maximize the number of seeps found across the campaign. This approach was able to find a novel seep site that was previously unknown to scientists that had studied the area, and to estimate the most important predictive features for the presence of seepage.

Search for Oceanic Hydrothermal Vents In volcanic areas, pressure driven by heat can cause fluids to release from beneath the sea floor in the form of hydrothermal vents [143]. Much like hydrocarbon seeps, these vents can only be detected at close range by autonomous or remotely operated underwater vehicles [101], and they are of interest to biologists, geologists, and volcanologists to provide insight into subsurface processes [63], their effects on the local biome [131], and possible threats to locals that may be posed by volcanic activity. Like in the search for hydrocarbon seeps, hydrothermal vents possess spatial correlations with other vents, and depend upon the depth and rugosity of local bathymetry.

This application demonstrated the need for adaptive sampling to be able to incorporate changes to the model and query as the mission proceeds. Query driven adaptive sampling was used in the search for hydrothermal vents in the Kolumbo caldera in November 2019 as a part of NASA's PSTAR project. A model was constructed to predict the presence of venting based on depth, rock type, presence to known vents, and multiple features derived from local bathymetry. Adaptive sampling produced plans for where to explore with remotely operated vehicles in order to maximize the number of vents found, and on separate dives, to maximize information about the distribution of vents. During the mission, encodings of sea floor characteristics were made available, and these were folded into Spock's environment model. The approach developed in this thesis was able to locate a previously unknown vent site that was not obvious from the bathymetry data alone.

Search for the Presence of Invasive Plant Life Park land and nature preserves ensure that animal and plant species native to an area can continue to thrive, and provide recreation to people who hike and camp in the area. Unfortunately, human presence in the area brings in invasive plant species, which can overwhelm local species if they are allowed to spread. Based on reports from park visitors, volunteer workers walk hiking trails and campsites, searching for evidence of invasive species and removing them when possible. However, the land is typically too vast for the number of volunteers, and search must be focused. Factors such as soil type, local

wind direction, local water flow, proximity to hiking trails, the presence of recent forest fires, and proximity to previously detected invasive species all are indicators of locations where an invasive species can take hold and spread.

This scenario demonstrated that query-driven adaptive sampling is useful not only for autonomous missions, but to guide human observers as well. The search for invasive plant life was considered as a case study for the technology developed in this thesis. Predictive models for the presence of invasive species based on the previously mentioned indicators were developed. No autonomous vehicles were planned to be used, but query driven adaptive sampling would be able to plan where volunteers should focus their efforts, in order to maximize the number of unwelcome invasive species that could be removed.

Threatened Species Population Estimation Rockfish are considered a threatened species in the Washington Puget Sound [148]. The state is interested in estimating and monitoring their population distribution and understanding the extent of suitable habitat for population recovery. Rockfish live in rocky terrain on the sea floor, but the distribution of suitable habitat is uncertain since low resolution sonar scans of the Sound do not fully distinguish whether certain locations can support rockfish. Data is gathered by lowering camera platforms and looking at the imagery they collect. The images help inform where rockfish are present, but also where habitat suitable for rockfish is present [100]. In order to construct accurate models of population and expected growth, Washington wishes to plan their observation dives to best understand the current rockfish population, and where additional possible habitat lies.

This mission demonstrated the need for environment models in adaptive sampling to be able to capture complex relationships between environment attributes. Rockfish population monitoring was used as a case study for adaptive sampling with queries. A model of bathymetric features must be determined based on position and the low resolution sonar imagery. Then, presence of rockfish is determined based on location and habitat suitability derived from bathymetry. Future observations would be se-

lected to maximize mutual information between observations and habitat suitability across the Sound, rockfish presence across the Sound, or a total estimate of the rockfish population, dependent on the query driving the mission. Even when the mission is motivated by rockfish estimates, observations focusing on the bathymetry may still be useful for determining overall rockfish population estimates.

Search for Water below the Lunar Surface Water ice has been found below the surface of the moon in craters that are permanently shadowed [115]. Understanding the distribution and thickness of that ice is critical for planning long duration manned lunar missions or lunar bases, because access to water will be required by the human crews [117]. A lunar rover can use a specialized drill to test for ice thickness, and by exploring the lunar surface it can test for overall prevalence. The path of the rover must periodically pass through illuminated areas so batteries on-board can be charged. In addition, traversing rough or highly sloped areas of the lunar surface subjects the rover to risk of damaging its wheels and becoming immobilized, but can lead to much faster paths between craters. The acceptable risk must be appropriately determined for the science return, and selections of where to observe must satisfy those bounds on risk.

The search for lunar ice was another demonstration of the need for risk bounded adaptive sampling techniques, and was considered as a case study for the work in this thesis. A model of lunar ice thickness would be constructed depending on location and fraction of time spent in shadow. Adaptive sampling planning would then be applied to choose locations to drill with an objective in order to maximize information about the distribution of ice thicknesses across the surface. The planner would select locations with paths between them that constrain the total risk of being immobilized based on the information that would be returned.

1.7 Summary

This thesis presents Spock; a system to perform query driven adaptive sampling. Spock plans missions of where to gather observations in order to best answer queries raised by users. In this thesis, we propose a query language to specify queries in adaptive sampling, and show how to use Monte Carlo planning to produce plans for any input query. Spock is designed to operate in risky, data-limited environments, so we present a method to plan with risk bounds that are functions of reward, propose a Gaussian process model to model environments with limited data in the presence of expert knowledge, and introduce a method to efficiently solve for structural relationships between environment variables. With these capabilities, Spock represents a significant leap in adaptive sampling for the uncertain, rapidly changing environments that humanity most wishes to explore.

Chapter 2

Adaptive Sampling with Queries

In this chapter, we consider the problem of solving for adaptive sampling strategies that best answer wide ranges of queries. We specify the intended problem to be solved by a query-driven adaptive sampling algorithm and the construction of a general-purpose query language to express the goals of a query-driven adaptive sampling planner.

Query-driven adaptive sampling allows a scientist to define what they wish to learn about or achieve in an adaptive sampling mission by specifying a *query*, and then constructs plans for where to gather observations to best answer those queries. Queries are designed to be expressive and capable of encoding a broad range of user intents. On a campaign exploring a basin with evidence of hydrocarbon seepage, example queries could include:

"Minimize the expected number of measurements so that the number of seeps found is at least 3",

which highlights that a query could focus on reaching a condition with the minimal number of measurements possible,

"Maximize the expected posterior mode of the number of seeps in the environment without backscatter above a specified threshold on the best 95% of mission outcomes",

which highlights that a query may only depend on certain outcomes of a mission,

"Maximize certainty in the effective length scale of spatial correlations in mound presence",

which highlights that a query may depend on model parameters like length scales of spatial correlations, and

"Maximize certainty in whether mounds are causally related to seeps",

which highlights that a query may depend on the existence of relationships between environment attributes like mounds and seeps.

We can see the flexibility required in query construction through the examples above. A query can focus on attributes at the locations that are actually observed, such as 'seeps found', or locations across the environment, including those that have not been observed, such as 'number of seeps in the environment'. Queries can focus on environment attributes such as 'seep presence', model parameters such as 'the length-scale of spatial correlations', correct choice of model such as 'whether mounds are causally related to seeps', or combinations of these such as 'seeps without backscatter above a specified threshold'. Furthermore, they can maximize expectations, posterior modes, or certainty, or minimize the number of measurements needed for any of those to reach a specific condition. Through query-driven adaptive sampling, we are able to capture and plan for all of these objectives.

Making adaptive sampling query-driven in this way provides two advantages to those interested in the gathered information. First, it allows missions to be planned with objectives that are useful, but were not expressible by any previous information-gathering planners. Second, a single planner is able to respond to the diverse set of allowed queries. This means that a scientist can encode and plan a mission to respond to a scientific question that was prompted by unexpected observations, and was not anticipated prior to a previous mission. The queries that can be input to existing approaches to adaptive sampling are not broad, and so considerations of a novel query would require development of a new algorithm, which would require a significant delay.

In this chapter we focus on a rich language for expressing adaptive sampling queries. We define our notion of a query as an objective, and a variable of interest computed as a function over environment variables, structural models, and parameters. Formulating a query this way allows users to construct queries from environment variables and connections between variables, as in the examples above, without needing to know the specifics of how the environment model generates predictions. Query-driven adaptive sampling asks the user to supply such a variable of interest and an appropriate objective, and an exploration policy of what to observe is solved that optimizes the objective applied to the variable of interest.

A key insight in this chapter is that providing this generality in queries makes it possible to construct objectives that lead to bias, causing an agent to deliberately avoid gathering new information. Our solution is to limit the space of query objectives to those that provably disallow this behavior.

2.1 Motivation

An example that has motivated the development of query-driven adaptive sampling is the search for natural underwater hydrocarbon seepage in oceanographic basins using autonomous underwater vehicles, which we refer to as the 'seep exploration scenario'. The presence of natural seeps is strong evidence for the presence of exploitable hydrocarbons within a basin, so there is commercial interest in locating evidence of seepage in order to justify more expensive subsurface studies. Bathymetric features on the sea floor such as mounds, pockmarks, faults, and combinations of those features can be identified by a ship-board sonar, and the strength of the 'acoustic backscatter' signal returned by a sonar pulse gives an indication of the material on the sea floor. This data is available before the deployment of any underwater vehicle, and it is known that these seafloor features are correlated with the presence of seepage, so they can be used to guide exploration. These locations act as a set of candidate seep sites that can be visited by the vehicle, and an algorithm is tasked with selecting the locations with appropriate combinations of features to visit to maximize the expected number

of seeps found.

Exploration within an environment is rarely constrained by a single objective, and is instead performed for a number of different reasons, each with distinct objectives, and requiring different observations to best answer. For example, beyond simply locating seeps, it is also of interest to estimate the total number of seeps within a basin, in order to inform estimates of the amount of hydrocarbons beneath the surface. Since natural seeps are unique biological habitats, biologists are also interested in these studies, but with an objective of determining the biomass that the seeps support. While each of the objectives in the seep exploration scenario involves gathering observations of the same variables (dissolved hydrocarbon concentrations, bathymetric features, evidence of bubbling, etc.) within the same environment, the optimal strategy of what measurements to take differs between objectives.

Algorithmically generated exploration strategies have seen success in satisfying some of these objectives. But each application has typically used a single algorithm that has been custom-built for the specific objective and environment being explored. Each distinct algorithm requires a long and difficult development process, alongside a dedicated expert in adaptive planning to do the design. With our approach, we aim to produce a single planning algorithm that can plan for the majority of queries that have been considered in the adaptive sampling literature to date, and add support for other queries that existing approaches cannot tackle.

A single algorithm with a highly focused query is undesirable in applications in which the environment is relatively unknown, such as the conditions encountered during oceanographic expeditions. In these cases, the data gathered is frequently unexpected and thought-provoking. In response to new data, scientists challenge their initial assumptions, formulate new hypotheses about the environment, and raise additional scientific questions that were not anticipated at the start of the expedition. For example, on one such seep exploration mission we conducted in December 2018, observations of differences in seep gas composition and intensity caused scientists to hypothesize that deep-sourced fluids were only present on one side of the basin, and they wished to test this hypothesis to further their understanding of the environment.

With query-driven adaptive sampling, a query testing this hypothesis could be input directly into the algorithm, and information could be gathered to answer it on the next dive. In contrast, custom-designed algorithms brought on the expedition are unsuitable for answering newly posed questions because the new objectives could not be predicted in advance. This limits the extent that answers to those questions can be pursued.

To avoid the expense of developing algorithms for multiple objectives, and to enable scientific exploration to answer questions that are prompted by data gathered during a mission, we propose an adaptive sampling approach that is flexible in its input. In our approach, known as query-driven adaptive sampling, a user formulates a scientific 'query' to be answered during a planned mission. The query specifies what the user wishes to understand, or an objective to be achieved. A single planning approach is then developed that is able to handle broad classes of queries.

We do not claim that our approach is as effective or efficient as algorithms that are specialized to specific environments and objectives, as those approaches are able to make use of problem-specific structure. Instead, our focus is on speed of response to a new query and generality. We show that our approach is able to handle the same objectives as many existing adaptive sampling algorithms, plus additional problems that are significantly beyond the capabilities of existing approaches. Using query-driven adaptive sampling, scientists on exploration missions are able to produce information-gathering plans automatically, without the difficulty of developing new algorithms in anticipation of the questions that will arise on a mission. Specialized algorithms may still be used when they are available, while our approach is intended to handle missions for which specialized algorithms have not been generated.

2.2 Related Work

At a high level, adaptive sampling refers to any procedure of experimental data gathering where the choice of variables to observe depends on the results of previous observations. In our motivating domain of underwater exploration, adaptive sam-

pling determines locations in space and time to measure with the instruments on an observation platform, typically an autonomous underwater vehicle or remotely operated vehicle, in order to best answer a scientific question about the environment. However, adaptive sampling is broadly applicable beyond environmental exploration, with applications ranging from medical trials [31] to human behavior surveys [134].

Within environmental exploration, adaptive sampling is typically performed in service of an objective that can be fit into one of two classes. The first class is concerned with improving the accuracy of variable(s) estimated from the environment. Examples include minimizing error in estimates of ocean temperature, salinity, and biological chemical concentrations [84], and locating chemical plumes [22, 102]. In the latter example, it may not be necessary to directly observe the source of the plume in order to precisely estimate its location. In contrast, the second class is concerned with maximizing the number of a certain type of observation or the magnitude of continuous observations like temperature that are taken by the observation platform. Examples include confirming high temperature measurements [10], observing sources of hydrocarbon seepage [144], and observing algal blooms [35]. The choice of whether to estimate a variable accurately or seek high value observations depends on the intentions of the mission designer, and we will allow both within query-driven adaptive sampling.

The methods deployed on robotic surveys are normally application-specific [62], and they do not easily generalize to objectives beyond those for which they were designed. Yet as previously mentioned, novel scientific questions may arise on prolonged exploration missions, and there may be opportunities for an observation platform to pursue a secondary objective. Existing adaptive sampling algorithms are not able to be used in these dynamic situations, as discussed in Section 2.1. Through query-driven adaptive sampling, we aim to provide a theoretically well-founded approach that is applicable to flexible objectives, including and beyond both the previously described classes.

2.2.1 Information Maximization

Mathematically justified approaches to improving accuracy of estimated variables frequently use an approach of maximizing mutual information between observed variables and variables of interest, which is equivalent to minimizing the expected posterior entropy of the objective variable. MacKay [92] derived strategies to maximize mutual information between observations and model parameters, environment variables in a region of interest, and the correct model, all using Taylor series approximations of information. Krause et al. [77] studied maximization of information about the value of a Gaussian process (GP) model, showing that it can be solved near-optimally with a greedy strategy, while Chen et al. [27] considered this strategy whenever observations are causally dependent on an objective variable. Since computation of mutual information is particularly easy for GP environments, a popular approach is to model exploration as a vehicle moving through a GP. Locations have been selected using dynamic programming [25, 90] or a genetic algorithm [60]. Our approach may be used to estimate mutual information for all these problems, while also allowing information to be maximized for more general variables of interest computed from model parameters or environment variables.

While mutual information is widely used, adaptive sampling has been formulated with other measures of information gain. These include average variance [55], average variance reduction [15, 48], estimator integrated mean squared error [157], and Fisher information [154]. Our approach is able to maximize information about the variable of interest considered in each of these cases, but we focus on mutual information for information maximization problems. To our knowledge, there have not been comprehensive studies on the relative performance of each measure of information.

2.2.2 Observable Maximization

The canonical abstraction for maximizing observations in the face of uncertainty is the multi-armed bandit problem. In the multi-armed bandit problem, sequential draws are taken from a fixed number of unknown probability distributions, with the

intention of maximizing the cumulative value of the draws [51]. Index policies that do not explicitly simulate returns are known to be able to achieve optimal return up to a constant factor [7, 47].

Adaptive sampling with an objective of maximizing observations over a continuous domain has many similarities with spatially correlated bandits [152] or the continuum armed bandit problem [71]. Stronger probabilistic lower bounds on reward are known in the case that the observed function is a Gaussian process [36, 129] than general continuum armed bandits. As a result, a GP model has been used in adaptive sampling to seek high luminosity observations [93] and hone in on hydrothermal hotspots [11], for example. Maximization of observations has also been studied for binary environment variables using nearest neighbor classifiers, under the name Bayesian active search [48, 64]. These methods make use of bounds on expectation to prune the space of observations, though the bounds are specific to model used.

Our approach is able to solve the same problem setups as the previously described observation-maximizing cases. However, many of these approaches make use of domain-specific bounds or heuristics, and that enables them to plan over longer or infinite horizons. Since the exact form of the bounds depends on the model and data being observed, and we cannot guarantee that the query to be maximized satisfies the assumptions for existing bounds, it follows that our approach is unable to recreate existing for infinite horizon strategies exactly. We instead maximize reward over a limited horizon, as in [10, 48]. While we lose application-specific performance, we gain the ability to maximize more general variables of interest, including those whose probability distributions are not known exactly, and add the ability to optimize over a subset of possible outcomes.

2.3 Problem Setup and Statement

In query-driven adaptive sampling, we wish to plan and execute a policy of actions that results in observations that let an agent learn about unknown variables. We will primarily focus on a mobile agent that can maneuver through an environment,

and collect local observations of the environment it is navigating. Before defining our problem, we first specify our assumptions on the environment and the capabilities of the agent. While our problem formulation is domain independent, we will motivate with the seep exploration scenario.

2.3.1 Environment Model Assumptions

The environment model assumptions described in this section specifies constraints on the model of the environment being used, in terms of a set of attribute variables and constraints on the relationships between these variables.

Consider, for example, the environment model for the seep exploration scenario. A vehicle must be sent to a series of selected sea floor locations in the hunt for hydrocarbon seepage. At each location, there is the presence or absence of bathymetric features like mounds and pockmarks, in addition to the presence or absence of an elevated backscatter signal. Without any of these signals, the probability of seepage is negligible, so candidate locations of study are restricted to a finite set of locations where at least one of these features is present. Each location then possesses a presence or absence of seepage, as well as other variables that can only be detected at close range like temperature, water column hydrocarbon concentrations, and the presence or absence of seep-supported fauna like clams.

Variables like the presence of seepage at different locations are spatially correlated, and complex statistical relationships exist between different variables at any given site. Experts are able to intuitively describe some of these relationships by saying, for example, that seeps cause elevated hydrocarbon concentrations, and that the presence of seeps is positively correlated with elevated backscatter. More complex details may be unknown, like the direction of causality between bathymetric features and seeps, and a numerical strength of the inter-variable correlations.

Next consider our specification of the environment model setup in general, which intends to capture all aspects of the seep exploration scenario. The environment model consists of a discrete set of locations $\mathcal{X}$ that the vehicle can reach and that are relevant for the variable of interest in the query. Our approach will predict the

variable of interest by predicting variables in the environment at these locations, so if a continuous space of observations is allowed, it must be discretized for use in our approach. A discrete domain assumption is not fundamentally different to other adaptive sampling or sensor placement problems [77, 90]. In the seep exploration scenario, each $\boldsymbol{x} \in \mathcal{X}$ is a location with at least one sea floor feature associated with seepage. In underwater missions, each location is specified in terms of longitude, latitude, depth, and optionally time. However, the locations may also be defined more abstractly; they may contain information such as sea floor slope or soil type that influences the behavior of the environment at those locations.

At each location, multiple *attributes*, which are detectable parts of the environment, are modeled. Example attributes include temperature, the presence or absence of a mound, or the type of coral species present at a location. Attributes are labeled with an index $m \in \{1, 2, ...\}$, so that we may refer to the presence of mounds as attribute 3, for example. We make no assumption on the form of the attributes, they may be continuously real valued or discrete, bounded or unbounded. We assume that each attribute exists at each location in $\mathcal{X}$.

We refer to the value of an attribute at a specific location as an *environment variable*, for example, hydrocarbon concentrations at a specific longitude, latitude, and depth. We refer to the vector of all environment variables, for all attributes and all locations in $\mathcal{X}$ as $\mathbf{y}$. The vector of environment variables of attribute m at all locations is written as $\mathbf{y}_m$. We refer to all environment variables at a location $\boldsymbol{x}$ by $\mathbf{y}(\boldsymbol{x})$ and a specific attribute as $\mathbf{y}_m(\boldsymbol{x})$.

In the environment model, environment variables of the *same* attribute at different locations are correlated, with correlations described by parameters $\mathbf{c}$. Inter-location correlations encode the fact that seeps appear in spatial clusters because they are fed by the same underground source, for example.

In addition to correlations between environment variables of the same attribute at different locations, there also exist correlations between different attributes. These correlations encode relationships like the fact that seepage is known to contribute to hydrocarbon concentrations and the existence of certain fauna nearby. Since not all

attributes are necessarily correlated, and it may be of interest to ask queries about which attributes directly affect others, we assume attributes are correlated following a graphical model. Certain queries may wish to ask about *causal* relationships between attributes, such as whether seeps generate mounds, or vice versa. In order to address queries about the direction of causality, we model attributes as being connected with directed edges. We further assume that attributes are correlated following a directed acyclic graph (DAG) model, so that queries about conditional independence between attributes can be easily modeled as queries about the existence of edges in a DAG. The disadvantage of this assumption is that it does not permit environments with cyclic causality to be modeled.

The DAG model is specified as $\mathcal{G} = (\{\mathbf{y}_m\}, \mathcal{E})$, where $\mathcal{E}$ is a set of directed edges $n \to m$ between attributes $\mathbf{y}_n$ and $\mathbf{y}_m$. Parameters $\boldsymbol{\lambda}$ describe the strength of the correlations between *different* attributes (whereas $\mathbf{c}$ described correlations between environment variables of the same attribute). We do not specify the specific form of the parameters $\mathbf{c}$ and $\boldsymbol{\lambda}$ or how they are used by the model, our discussion here is intended to highlight the parameters which can be incorporated into user defined queries.

Using parent sets $\Pi_m^{\mathcal{G}} = \{n \mid (n \to m) \in \mathcal{E}\}$, the DAG structure implies that the environment variables factor as

$$p(\mathbf{y} \mid \mathbf{c}, \mathcal{E}, \boldsymbol{\lambda}) = \prod_m p(\mathbf{y}_m \mid \mathbf{y}_{\Pi_m^{\mathcal{G}}}, \mathbf{c}_m, \boldsymbol{\lambda}_{m,\Pi_m^{\mathcal{G}}}), \tag{2.1}$$

where $\mathbf{y}_{\Pi_m^{\mathcal{G}}} = \{\mathbf{y}_n \mid (n \to m) \in \mathcal{E}\}$, $\mathbf{c}_m$ are spatial correlation parameters associated with attribute m, and $\boldsymbol{\lambda}_{m,\Pi_m^{\mathcal{G}}}$ are edge parameters associated with the edges from $\mathbf{y}_{\Pi_m^{\mathcal{G}}}$ to $\mathbf{y}_m$. The use of a DAG model of the environment allows the user to pose queries about dependence or independence between different attributes, for example by asking whether a sequence of edges exists that connects the presence of seepage to the presence of mounds, and pose queries about causal relationships between variables, for example by asking whether the presence of mounds is a descendant of (and therefore a causal consequence of) the presence of seeps.

It is not necessary to have a single exact model with fixed parameters. Queries may ask about the distributions over parameters, for example. Instead, the environment model $\mathscr{M} = \langle \mathbf{y}, \mathbf{c}, \mathcal{E}, \boldsymbol{\lambda} \rangle$ is considered to be a random variable that contains the parameters necessary to make predictions in the environment model, in addition to values of environment variables, so that variation in the parameters is allowed. When conditioned on a set of prior observations $\mathbf{o}_0$ derived from some environment variables, the environment model uses the values of the parameters $\mathbf{c}$, $\mathcal{E}$, and $\boldsymbol{\lambda}$ to provide predictions of $\mathbf{y}$. That is,

$$p(\mathscr{M} \mid \mathbf{o}_0) = p(\mathbf{y} \mid \mathbf{c}, \mathcal{E}, \boldsymbol{\lambda}, \mathbf{o}_0)\ p(\mathbf{c}, \mathcal{E}, \boldsymbol{\lambda} \mid \mathbf{o}_0). \tag{2.2}$$

Beyond the requirements listed in this section, the only restrictions we place on the model is that after a training procedure, the distribution $p(\mathscr{M} \mid \mathbf{o}_0)$ can be sampled from in an efficient manner. One candidate model that fulfills all our requirements is the heterogeneous AcyGP model, which we will use for all our experiments. Our query-driven sampling approach is separate from the model it acts on, and any model that can be sampled from will be sufficient. Models that do not implement all the features above, such as a lack of causal structure or spatial correlation parameters, can be used, without allowing queries that focus on those parts of the model.

2.3.2 Agent Assumptions

The agent assumptions described in this section specify constraints on the motion of the agent and what it is able to observe.

In the seep exploration scenario, an autonomous underwater vehicle moves between candidate seep sites, stopping to take measurements at each. There is only enough time in a mission to visit and observe a specified number of sites, since observation can be an time intensive process. Additionally, battery constraints on the vehicle can dictate how far the vehicle can move over the course of the mission.

When the vehicle takes measurements at a location $\boldsymbol{x}$, it is not guaranteed to be able to measure all environment variables $\mathbf{y}(\boldsymbol{x})$, and there is expected to be some

noise. For instance, a vehicle without a mass spectrometer may be unable to measure hydrocarbon concentrations, and whether a seep can be considered active is sometimes ambiguous. The following agent specification captures restrictions on motion and the noise in observations.

We model the agent at discrete time steps with index $t \in \{0, 1, \ldots, T\}$, at which the agent occupies some location $\boldsymbol{x}_t \in \mathcal{X}$. The agent begins at location $\boldsymbol{x}_0$ with a vector of prior observations $\mathbf{o}_0$, and upon reaching location $\boldsymbol{x}_t$, it takes a measurement $\mathbf{o}_t$, which is based on the true value of the environment variables $\mathbf{y}(\boldsymbol{x}_t)$. In general, the measurement may be noisy, measure only a subset of environment variables, or output a processed data product. To account for these possibilities, we assume the existence of a distribution over measurements $p(\mathbf{o}_t \mid \mathbf{y}(\boldsymbol{x}_t))$. Over t time steps, the measurements are collected into a vector $\mathbf{o}_{1:t}$. We refer to the space of measurement vectors of different possible length missions (including $\mathbf{o}_1, \mathbf{o}_{1:2}, \ldots, \mathbf{o}_{1:T}$) as $\mathcal{O}$. We do not make assumptions on the form of the elements of $\mathbf{o}_t$ relative to the environment variables. For example, a discrete quantization may be taken from a continuous environment variable.

At each time step, we decide on an action for the vehicle to take that will change its location. We assume that a finite set of actions $\mathcal{A}$ are available at each time step. The agent dynamics model $\boldsymbol{x}_{t+1} = d(\boldsymbol{x}_t, a)$ specifies how the state of the agent changes from $\boldsymbol{x}_t$ in response to action $a \in \mathcal{A}$. In the seep exploration scenario, this may simply be the set of candidate locations that have not yet been visited but are within battery range. In this chapter, we assume a deterministic dynamics model with discrete time.

2.3.3 Query Assumptions

The query assumptions described in this section provide an overview of the expected structure of a query and intuitions behind those choices. A complete discussion of the query language can be found in Section 2.7.

A query $\mathcal{Q} = \langle f_{\mathcal{Q}}, J_{\mathcal{Q}}, \Delta_{\mathcal{Q}} \rangle$ defines two functions and an optional threshold. The first component is the *query function* $f_{\mathcal{Q}}$, which outputs a scalar variable of interest ζ

for the problem. Queries include a query function to specify what should be studied, or what a scientist wishes to know about. It is computed from the locations the observation platform visits and a possible realization of the environment, including model parameters and the true values of any environment variables. Intuitive examples include the maximum value of an attribute $\mathbf{y}_m$ over $\mathcal{X}$, or the strength of the correlation between two attributes in the environment.

While the query function states *what* should be studied, the *query objective* $J_{\mathcal{Q}}$ describes *how* ζ should be studied. Intuitively, it states whether users are interested in the uncertainty of ζ, the mode of ζ, or finding high values of ζ, and in this way it acts as a generalization of previously identified classes of adaptive sampling problems. Example query objectives include computing the information between ζ and $\mathbf{o}_{1:T}$, or the total expected value of ζ.

Finally, the *query threshold* $\Delta_{\mathcal{Q}}$, distinguishes between two qualitatively different types of missions; one where the observing agent gathers as much information as possible in the time it has available, and another where the mission is run until a goal is met. Both are useful ways of performing adaptive sampling, and are applicable in different types of scenarios. The query threshold indicates that a the latter type of mission is desired, and encodes a quantity of the query objective that is sufficient for the mission, after which the mission can end. Examples include a sufficient level of information, or an expected level of certainty in the answer to a query.

2.3.4 Problem Statement

We now define our problem formally, and then introduce each part in detail. A user supplies a query over the variables in the environment model, and our approach solves for a series of actions to be taken by a vehicle that optimizes an objective that depends on its observations and the query output.

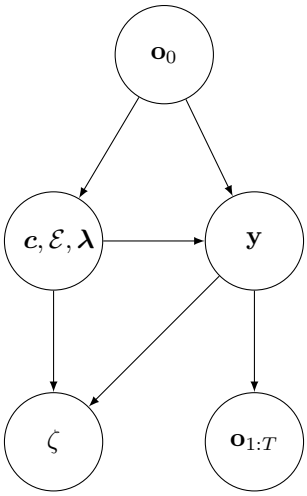


Figure 2-1: Graphical model of relationships between environment and query variables.

Problem 1. *Determine a policy* $\pi : \mathcal{O} \to \mathcal{A}$ *that optimizes*

$$\begin{aligned} \max_\pi \quad & W(J_\mathcal{Q}, \Delta_\mathcal{Q}) \\ \text{s.t.} \quad & \zeta = f_\mathcal{Q}(\boldsymbol{x}_{1:T}, \mathscr{M}) \\ & \boldsymbol{x}_{t+1} = d(\boldsymbol{x}_t, \pi(\mathbf{o}_{1:t})) \\ & \mathbf{o}_{t+1} \sim p(\mathbf{o}_{t+1} \mid \mathbf{y}(\boldsymbol{x}_{t+1})) \end{aligned}$$

where $\mathcal{Q} = \langle f_\mathcal{Q}, J_\mathcal{Q}, \Delta_\mathcal{Q} \rangle$ *is a query to be solved,* $\mathscr{M} \sim p(\mathscr{M})$ *is a description of the environment model, and* $d : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$ *is a vehicle dynamics model.*

The top level objective W is a quantity computed from the query objective specification and the query threshold. In most cases, $W(J_\mathcal{Q}, \Delta_\mathcal{Q}) = J_\mathcal{Q}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$, but we will also specify variable length missions, where the objective is to minimize the number of observations needed to reach the query threshold. We distinguish W from $J_\mathcal{Q}$ because it will be necessary to estimate $J_\mathcal{Q}$ in order to construct an estimate for W. Given the query, the policy is constructed to lead to observations that will optimize the top level objective.

A graphical model of relationships between variables in our problem is given in Figure 2-1. Through appropriate selection of ζ to be elements of $\mathbf{y}$, $\mathbf{c}$, $\mathcal{E}$, or $\boldsymbol{\lambda}$, our formulation of queries encompasses most existing work on adaptive sampling.

2.3.5 Examples of Query-Driven Adaptive Sampling Problems

Problem 1 is more general than existing adaptive sampling problems. In order to illustrate grounded examples of query-driven adaptive sampling problems, and to demonstrate the richness of our framework, we now show how existing adaptive sampling problems may be formulated in our query-driven framework.

Information Maximization in a Gaussian Process

The most common adaptive sampling problem considers an agent navigating an environment described by a single-output Gaussian process model [77, 90]. The agent is tasked with maximizing the information between its noisy observations of the environment and the true value of the environment variables at those locations. A single-output Gaussian process model can be described with a single attribute environment, resulting in a DAG structure with a single variable $\mathbf{y}$. The set of edges and edge parameters are then empty, while $\mathbf{c}$ consists of deterministic covariance kernel parameters in the Gaussian process. Predictions of the environment are then achieved as

$$p(\mathscr{M} \mid \mathbf{o}_0) = p(\mathbf{y} \mid \mathbf{c}, \mathbf{o}_0).$$

The variable of interest in this problem is the true value of the environment variables at the locations visited, so that

$$f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \mathscr{M}) = \mathbf{y}(\boldsymbol{x}_{1:T}).$$

Finally, the objective is to maximize mutual information between observations and environment variables in the environment, which is achieved through a query objective of mutual information I,

$$J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = I(\zeta; \mathbf{o}_{1:T} \mid \mathbf{o}_0).$$

In this work we focus on scalar variables of interest, which facilitates solving for policies for certain complex objectives. Nevertheless, query definitions can be considered

separately from the solution method, and it is still meaningful to formulate a query with vector ζ like in this example. Furthermore, only certain objectives require scalar ζ, and a query that maximizes mutual information of vector ζ like in this case is fully compatible with our framework.

Bayesian Active Search

Alternatively, Bayesian active search [48, 64] seeks to maximize the expected number of positive observations in a binary field. In this case, the environment models a single attribute that is either 0 or 1, with

$$f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \mathscr{M}) = \sum_{t=1}^{T} \mathbf{y}(\boldsymbol{x}_t)$$
$$J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}\left[\zeta \mid \mathbf{o}_0\right].$$

Our formulation is able to generalize these problems by using more complex environment models including environment models with more than a single attribute, considering more general functions of the observations and environment, or using a different objective.

2.4 Overview of Approach

To describe query-driven adaptive sampling, we will first describe our model of a query. We define a query in terms of a function supplied by the user that defines a variable of interest, an objective, and an optional sufficient condition. The goal of query-driven adaptive sampling will be to maximize the objective applied to the variable of interest if no sufficient condition is applied, or minimize the number of observations for the objective to reach the sufficient condition if it is given. Query objectives fall into one of three classes, and the range of possible objectives can be extended through transformations that can be applied to them. We refer to these transformations as prior and posterior specializations, which respectively constrain the values of observations and values of the variable of interest that are considered in

the query.

This chapter is structured as follows. Sections 2.5 and 2.6 review mutual information and sample-based estimators respectively, which will be used in the development of query-driven adaptive sampling. Section 2.3 formalizes the problem statement behind query-driven adaptive sampling. Section 2.7 defines queries formally, while 2.8 describes why certain limitations on query definitions were taken by proving that certain queries can lead to unscientific behavior.

2.5 Preliminaries: Definition of Mutual Information

In this section, we review preliminaries that are necessary to understand how certain query objectives are formulated in query-driven adaptive sampling. Adaptive sampling is often performed with an objective to maximize mutual information between observations taken by the agent and all environment variables. In query-driven adaptive sampling, we allow more general objectives, but include mutual information as an important case. We now review how mutual information is defined, in order to make clear how mutual information is an expectation of log probabilities. This definition is used to clarify the similarity between mutual information and other objectives.

Consider two random variables $\boldsymbol{x} \sim p(\boldsymbol{x})$ and $\boldsymbol{y} \sim p(\boldsymbol{y})$ defined over domains $\mathcal{X}$ and $\mathcal{Y}$. Here, boldface notation indicates that $\boldsymbol{x}$ and/or $\boldsymbol{y}$ may be vector-valued random variables. The following discussion applies equally to discrete random variables by replacing integrals with sums.

Mutual information quantifies a measure of the dependence between $\boldsymbol{x}$ and $\boldsymbol{y}$, or how much measuring $\boldsymbol{y}$ tells us about the true value of $\boldsymbol{x}$. Specifically, it measures the expected change from the distribution $p(\boldsymbol{x})$ to the posterior distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$ obtained by measuring $\boldsymbol{y}$ in terms of bits or nats, dependent on whether the logarithm is base 2 or base e respectively. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent, observation of $\boldsymbol{y}$ does not impact $\boldsymbol{x}$, since $p(\boldsymbol{x}) = p(\boldsymbol{x} \mid \boldsymbol{y})$. $\boldsymbol{y}$ contains no information about $\boldsymbol{x}$ in this case, and mutual information is zero. Conversely, if $\boldsymbol{x}$ is strongly dependent on $\boldsymbol{y}$, then $p(\boldsymbol{x} \mid \boldsymbol{y})$ has a very different form than $p(\boldsymbol{x})$, and depends upon the specific value of

$\boldsymbol{y}$ observed. Observing $\boldsymbol{y}$ therefore reveals a large amount of information about the value of $\boldsymbol{x}$, and so the mutual information is high.

Mutual information may be formally introduced in terms of Kullback-Leibler divergence [78]. The Kullback-Leibler divergence (KL divergence) is a directed distance in the space of probability distributions, so that KL divergence from a distribution with density function $q(\boldsymbol{x})$ to a distribution with density function $p(\boldsymbol{x})$ increases as $q(\boldsymbol{x})$ and $p(\boldsymbol{x})$ become increasingly different. It is a measurement of how much information results from changing belief from prior $q(\boldsymbol{x})$ to posterior $p(\boldsymbol{x})$. More intuitively, the Kullback-Leibler divergence (KL divergence) may be thought of as a measure of how much one distribution differs from another. The KL divergence from $q(\boldsymbol{x})$ to $p(\boldsymbol{x})$ is defined as

$$D_{KL}(p(\boldsymbol{x}) \,||\, q(\boldsymbol{x})) := \int_{\mathcal{X}} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \, d\boldsymbol{x}, \tag{2.3}$$

with $0 \log 0/q(\boldsymbol{x})$ interpreted as 0. To ensure that KL divergence is finite, we require that $p(\boldsymbol{x}) = 0$ wherever $q(\boldsymbol{x}) = 0$. It can be shown that $D_{KL}(p(\boldsymbol{x}) \,||\, q(\boldsymbol{x})) \geq 0$, and equals 0 if and only if $p(\boldsymbol{x}) = q(\boldsymbol{x})$, which assists in interpretation as a distance.

The fact that KL divergence is directed means that $D_{KL}(p(\boldsymbol{x}) \,||\, q(\boldsymbol{x}))$ is not generally equal to $D_{KL}(q(\boldsymbol{x}) \,||\, p(\boldsymbol{x}))$. It typically requires stronger evidence to move from a prior with high certainty to an uncertain posterior, than to refine a prior with low certainty into a higher certainty posterior.

In these terms, the mutual information between $\boldsymbol{x}$ and $\boldsymbol{y}$, written as $I(\boldsymbol{x}; \boldsymbol{y})$ [32], may be considered to be the expectation of the directed distance from $p(\boldsymbol{x})$ to $p(\boldsymbol{x} \mid \boldsymbol{y})$, with the expectation taken with respect to $\boldsymbol{y}$. In this way, mutual information encodes the expected change in $\boldsymbol{x}$ after conditioning on $\boldsymbol{y}$.

$$\begin{aligned} I(\boldsymbol{x}; \boldsymbol{y}) &:= \mathbb{E}_{\boldsymbol{y}} \left[D_{KL}(p(\boldsymbol{x} \mid \boldsymbol{y}) \,||\, p(\boldsymbol{x})) \right] \\ &= \int_{\mathcal{X},\mathcal{Y}} p(\boldsymbol{x} \mid \boldsymbol{y}) p(\boldsymbol{y}) \log \frac{p(\boldsymbol{x} \mid \boldsymbol{y})}{p(\boldsymbol{x})} \, d\boldsymbol{x} \, d\boldsymbol{y} \\ &= \int_{\mathcal{X},\mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y}) \log \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p(\boldsymbol{x}) \, p(\boldsymbol{y})} \, d\boldsymbol{x} \, d\boldsymbol{y}. \end{aligned} \tag{2.4}$$

We may also introduce the (specific) conditional mutual information, conditioned on a specific outcome $\boldsymbol{z}$ as

$$\begin{aligned} I(\boldsymbol{x};\boldsymbol{y} \mid \boldsymbol{z}) &:= \mathbb{E}_{\boldsymbol{y}}\left[D_{KL}(p(\boldsymbol{x} \mid \boldsymbol{y},\boldsymbol{z}) \,||\, p(\boldsymbol{x} \mid \boldsymbol{z}))\right] \\ &= \int_{\mathcal{X},\mathcal{Y}} p(\boldsymbol{x},\boldsymbol{y} \mid \boldsymbol{z}) \log \frac{p(\boldsymbol{x},\boldsymbol{y} \mid \boldsymbol{z})}{p(\boldsymbol{x} \mid \boldsymbol{z})\ p(\boldsymbol{y} \mid \boldsymbol{z})}\ d\boldsymbol{x}\ d\boldsymbol{y}. \end{aligned} \tag{2.5}$$

We refer to non-specific mutual information that averages over outcomes of $\boldsymbol{z}$ as an explicit expectation $\mathbb{E}_{\boldsymbol{z}}\left[I(\boldsymbol{x};\boldsymbol{y} \mid \boldsymbol{z})\right]$.

When handling mutual information in the development of query-driven adaptive sampling, we have to frequently compute terms of the form $\log p(\boldsymbol{x},\boldsymbol{y} \mid \boldsymbol{z})/p(\boldsymbol{x} \mid \boldsymbol{z})\ p(\boldsymbol{y} \mid \boldsymbol{z})$. For brevity, we refer to this quantity as the *log probability ratio* lr, where

$$lr(\boldsymbol{x};\boldsymbol{y} \mid \boldsymbol{z}) := \log \frac{p(\boldsymbol{x},\boldsymbol{y} \mid \boldsymbol{z})}{p(\boldsymbol{x} \mid \boldsymbol{z})\ p(\boldsymbol{y} \mid \boldsymbol{z})}, \tag{2.6}$$

so that $I(\boldsymbol{x};\boldsymbol{y} \mid \boldsymbol{z}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\left[lr(\boldsymbol{x};\boldsymbol{y} \mid \boldsymbol{z})\right]$. This expectation will allow us to relate mutual information to the expectations of other quantities when defining queries.

From (2.5) it can be seen that $I(\boldsymbol{x};\boldsymbol{y} \mid \boldsymbol{z}) = I(\boldsymbol{y};\boldsymbol{x} \mid \boldsymbol{z})$. This means that the information about $\boldsymbol{x}$ contained in $\boldsymbol{y}$ matches the information contained in $\boldsymbol{y}$ about $\boldsymbol{x}$, and $I(\boldsymbol{x};\boldsymbol{y} \mid \boldsymbol{z})$ is a measure of the dependence between $\boldsymbol{x}$ and $\boldsymbol{y}$ when conditioned on $\boldsymbol{z}$. Though mutual information *is* the expectation of KL divergence and KL divergence is a directed measure, the expectation of KL divergence does not depend on direction.

The fact that mutual information is symmetric may be intuitively understood in the following way. Any statement in which knowledge of $\boldsymbol{y}$ changes understanding of $\boldsymbol{x}$ also results in a statement in which knowledge of $\boldsymbol{x}$ changes understanding of $\boldsymbol{y}$. For instance, if we know that $\boldsymbol{y} = 0 \implies \boldsymbol{x} = 0$, then we also know that $\boldsymbol{x} \neq 0 \implies \boldsymbol{y} \neq 0$. At first glance, the consequence of the first statement seems stronger, especially if we initially believed that $\boldsymbol{x} = 0$ was a low probability event. But in that case, it should be understood that the condition $\boldsymbol{x} \neq 0$ also applies more frequently, so that the change in uncertainty averaged across outcomes of $\boldsymbol{x}$ and $\boldsymbol{y}$

due to either rule is the same. A basic introduction to mutual information is provided by Cover [32].

An alternative way to introduce mutual information is in terms of Shannon entropy [123]. We introduce these forms because they will be useful in the derivations of estimators for mutual information. The specific conditional Shannon entropy of a random variable $\boldsymbol{x}$ conditioned on outcome $\boldsymbol{z}$ encodes a measure of uncertainty in the distribution $p(\boldsymbol{x} \mid \boldsymbol{z})$, and is defined as

$$H(\boldsymbol{x} \mid \boldsymbol{z}) := -\int_{\mathcal{X}} p(\boldsymbol{x} \mid \boldsymbol{z}) \log p(\boldsymbol{x} \mid \boldsymbol{z})\, d\boldsymbol{x}. \tag{2.7}$$

Mutual information may equivalently be thought of as the expected reduction in entropy in $\boldsymbol{x}$ caused by measuring $\boldsymbol{y}$, as

$$\begin{aligned} I(\boldsymbol{x}; \boldsymbol{y} \mid \boldsymbol{z}) &= \int_{\mathcal{X},\mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}) \log \frac{p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z})}{p(\boldsymbol{x} \mid \boldsymbol{z})\ p(\boldsymbol{y} \mid \boldsymbol{z})}\ d\boldsymbol{x}\ d\boldsymbol{y} \\ &= -\int_{\mathcal{X}} p(\boldsymbol{x} \mid \boldsymbol{z}) \log p(\boldsymbol{x} \mid \boldsymbol{z})\ d\boldsymbol{x} + \int_{\mathcal{X},\mathcal{Y}} p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z})\ p(\boldsymbol{y} \mid \boldsymbol{z}) \log p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) \\ &= H(\boldsymbol{x} \mid \boldsymbol{z}) - \mathbb{E}_{\boldsymbol{y}|\boldsymbol{z}} \left[H(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) \mid \boldsymbol{z} \right]. \end{aligned} \tag{2.8}$$

We can also express this formula as the difference between joint entropy and marginal entropy, as

$$\begin{aligned} I(\boldsymbol{x}; \boldsymbol{y} \mid \boldsymbol{z}) &= \int_{\mathcal{X},\mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}) \log \frac{p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z})}{p(\boldsymbol{x} \mid \boldsymbol{z})\ p(\boldsymbol{y} \mid \boldsymbol{z})}\ d\boldsymbol{x}\ d\boldsymbol{y} \\ &= -\int_{\mathcal{X}} p(\boldsymbol{x} \mid \boldsymbol{z}) \log p(\boldsymbol{x} \mid \boldsymbol{z})\ d\boldsymbol{x} - \int_{\mathcal{Y}} p(\boldsymbol{y} \mid \boldsymbol{z}) \log p(\boldsymbol{y} \mid \boldsymbol{z})\ d\boldsymbol{y} \\ &\quad + \int_{\mathcal{X},\mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}) \log p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}) \\ &= H(\boldsymbol{x} \mid \boldsymbol{z}) + H(\boldsymbol{y} \mid \boldsymbol{z}) - H(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}). \end{aligned} \tag{2.9}$$

Formulae (2.8) and (2.9) will be used to introduce sample-based mutual information estimators, which will be used to estimate information within query-driven adaptive sampling.

Finally, it will also be useful to note that log probability ratio may be expressed as the following sum, which we will use when considering incremental increases in log probability ratio due to multiple observations,

$$\begin{aligned} lr(\boldsymbol{x}; \boldsymbol{y}_1, \boldsymbol{y}_2 \mid \boldsymbol{z}) &= \log \frac{p(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2 \mid \boldsymbol{z})}{p(\boldsymbol{x} \mid \boldsymbol{z})\ p(\boldsymbol{y}_1, \boldsymbol{y}_2 \mid \boldsymbol{z})} \\ &= \log \frac{p(\boldsymbol{y}_2 \mid \boldsymbol{z})\ p(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2 \mid \boldsymbol{z})}{p(\boldsymbol{y}_1, \boldsymbol{y}_2 \mid \boldsymbol{z})\ p(\boldsymbol{x}, \boldsymbol{y}_2 \mid \boldsymbol{z})} + \log \frac{p(\boldsymbol{x}, \boldsymbol{y}_2 \mid \boldsymbol{z})}{p(\boldsymbol{x} \mid \boldsymbol{z})\ p(\boldsymbol{y}_2 \mid \boldsymbol{z})} \\ &= \log \frac{p(\boldsymbol{x}, \boldsymbol{y}_1 \mid \boldsymbol{y}_2, \boldsymbol{z})}{p(\boldsymbol{y}_1 \mid \boldsymbol{y}_2, \boldsymbol{z})\ p(\boldsymbol{x} \mid \boldsymbol{y}_2, \boldsymbol{z})} + \log \frac{p(\boldsymbol{x}, \boldsymbol{y}_2 \mid \boldsymbol{z})}{p(\boldsymbol{x} \mid \boldsymbol{z})\ p(\boldsymbol{y}_2 \mid \boldsymbol{z})} \\ &= lr(\boldsymbol{x}; \boldsymbol{y}_1 \mid \boldsymbol{y}_2, \boldsymbol{z}) + lr(\boldsymbol{x}; \boldsymbol{y}_2 \mid \boldsymbol{z}). \end{aligned} \tag{2.10}$$

This leads to the chain rule of mutual information, which will be used when replanning in response to new measurements.

$$I(\boldsymbol{x}; \boldsymbol{y}_1, \boldsymbol{y}_2 \mid \boldsymbol{z}) = \mathbb{E}_{\boldsymbol{y}_2 \mid \boldsymbol{z}} \left[I(\boldsymbol{x}; \boldsymbol{y}_1 \mid \boldsymbol{y}_2, \boldsymbol{z}) \mid \boldsymbol{z} \right] + I(\boldsymbol{x}; \boldsymbol{y}_2 \mid \boldsymbol{z}). \tag{2.11}$$

2.6 Preliminaries: Sample-Based Density and Information Estimators

In this section, we review preliminaries that are necessary to understand how reward is computed within query-driven adaptive sampling. To develop query-driven adaptive sampling, we require estimates of quantities of the form $\mathbb{E}[p(\boldsymbol{x})]$ and $\mathbb{E}[\log p(\boldsymbol{x})]$, both over all $\boldsymbol{x} \in \mathcal{X}$ and subsets of $\mathcal{X}$. When the form of $p(\boldsymbol{x})$ is unknown but it can be sampled, Monte Carlo sampling allows for estimation of $\mathbb{E}[f(\boldsymbol{x})]$ by averaging $f(\boldsymbol{x})$ evaluated at samples drawn from $p(\boldsymbol{x})$. However, $\mathbb{E}[p(\boldsymbol{x})]$ and $\mathbb{E}[\log p(\boldsymbol{x})]$ depend explicitly on $p(\boldsymbol{x})$, and so explicit estimation of $p(\boldsymbol{x})$ or $\log p(\boldsymbol{x})$ is required. In this section, we review estimators for probability density and mutual information based on a finite set of samples $\mathcal{D}$. These estimators are used to compute the value of missions when queries are expressed in terms of probabilities of certain outcomes or information gain. In this section and throughout this chapter, we use a hat to refer to an estimate of a quantity, so that $\hat{p}$ is an estimate for the true value of a variable

p.

2.6.1 Kernel Density Estimation and k-NN Density Estimation

In order to reason over queries that consider the probabilities of certain events, either continuous or discrete, we require estimation of probability mass and probability density. For a random variable $\boldsymbol{x}$ with a discrete domain and a set of samples $\mathcal{D} = \{\boldsymbol{x}^{(j)}\}_{j=1}^{L}$ drawn independently from $p(\boldsymbol{x})$, it is well known that an estimator $\hat{p}(\boldsymbol{x})$ for $p(\boldsymbol{x})$ can be obtained as the ratio of $L_{\boldsymbol{x}}$, the number of samples for which $\boldsymbol{x}^{(i)} = \boldsymbol{x}$, and the total number of samples L,

$$\hat{p}(\boldsymbol{x}) = \frac{L_{\boldsymbol{x}}}{L}. \tag{2.12}$$

It is also well known that this estimator is unbiased for any finite L and consistent, which follows directly from properties of a binomial distribution.

For continuous $\boldsymbol{x} \in \mathbb{R}^{D_{\boldsymbol{x}}}$ with probability density $p(\boldsymbol{x})$, a finite set of samples will never sample every possible outcome of $\boldsymbol{x}$, so we cannot use (2.12) naively. Instead, specialized estimators for probability density include kernel density estimators [120, 125] and k-nearest neighbor (k-NN) density estimators [88], that can estimate non-zero probability density for $\boldsymbol{x}$ outside the set of samples. Kernel density estimators estimate density using a kernel function K that integrates to 1 as

$$\hat{p}_{kern}(\boldsymbol{x}) = \frac{1}{L\ h^{D_{\boldsymbol{x}}}} \sum_{i=1}^{L} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}^{(i)}}{h}\right). \tag{2.13}$$

for a scalar *bandwidth* h that is chosen to increase as a function of L. The intuition behind (2.13) is that the kernel K places probability mass near samples $\boldsymbol{x}^{(i)}$. Examples of kernels include step functions or Gaussian density functions centered on $\boldsymbol{x}^{(i)}$, and the width of those functions is determined by the bandwidth h. For example, a large value of h would place a wide Gaussian density centered at each $\boldsymbol{x}^{(i)}$, while a small value of h would place a thin Gaussian density centered at each $\boldsymbol{x}^{(i)}$. Parzen [105]

showed that the estimator is asymptotically unbiased and consistent for univariate distributions, while Cacoullos [21] showed that this remains true for multivariate data for any $\boldsymbol{x}$, where $p(\boldsymbol{x})$ is continuous under the assumptions that

$$\begin{aligned}
\lim_{L\to\infty} h(L) &= 0 \\
K(\boldsymbol{x}) &\geq 0 \\
\sup_{\boldsymbol{x}} |K(\boldsymbol{x})| &< \infty \\
\int |K(\boldsymbol{x})|\, dx &= 1 \\
\lim_{|\boldsymbol{x}|\to\infty} |\boldsymbol{x}|^{D_{\boldsymbol{x}}}\, K(\boldsymbol{x}) &= 0 \\
\forall \boldsymbol{x}, K(\boldsymbol{x}) &= K(-\boldsymbol{x}),
\end{aligned}$$

though these are stronger assumptions than technically needed for asymptotic unbiasedness and consistency [50]. While the estimator is asymptotically unbiased, for finite L the estimator is biased in general, meaning that we cannot assume that the average of the estimator equals the true probability density on average across outcomes. Under the further assumptions that $\lim_{L\to\infty} L\ h(L)^{2D_{\boldsymbol{x}}} = \infty$, so that $h(L)$ shrinks sufficiently slowly, and that the Fourier transform of $K(\boldsymbol{x})$ is absolutely integrable, the estimator is also consistent.

While kernel density estimators provide flexibility in the choice of kernel and bandwidth, the choice of bandwidth influences the convergence rate of the estimator. A large field of work focuses on deriving optimal bandwidths in terms of convergence rate for specific choices of kernels and classes of densities [59, 66], but this requires some knowledge of the density being estimated. An alternative approach is to choose the bandwidth h based on some pre-processing of the sample set. k-NN density estimators do this by changing the estimates according to the distance to the k-th nearest neighbor of the point $\boldsymbol{x}$.

Denote the distance from $\boldsymbol{x}$ to the k-th nearest sample as $\varepsilon^{\boldsymbol{x}}$. The distance may be measured with respect to any norm, and for numerical stability and ease of computation, the infinity norm is often used, where $\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\|_\infty = \max_k |x_k^{(i)} - x_k^{(j)}|$. Then

let $\mathscr{B}(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}})$ denote the open ball of radius $\varepsilon^{\boldsymbol{x}}$ around point $\boldsymbol{x}$, that is $\mathscr{B}(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}}) = \{\boldsymbol{x}' \mid \|\boldsymbol{x}' - \boldsymbol{x}\| < \varepsilon^{\boldsymbol{x}}\}$. The ball is defined with respect to the same norm as distance, so for the infinity norm, is a cube with side length 2 $\varepsilon^{\boldsymbol{x}}$. Let $P(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}})$ be the probability mass inside $\mathscr{B}(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}})$, and $V_{\boldsymbol{x}}(\varepsilon^{\boldsymbol{x}})$ be the volume of $\mathscr{B}(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}})$,

$$P(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}}) = \int_{\mathscr{B}(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}})} p(\boldsymbol{x}) \; d\boldsymbol{x} \tag{2.14}$$

$$V_{\boldsymbol{x}}(\varepsilon^{\boldsymbol{x}}) = \int_{\mathscr{B}(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}})} d\boldsymbol{x}. \tag{2.15}$$

The k-NN density estimator makes the approximations $p(\boldsymbol{x}) \approx P(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}})/V_{\boldsymbol{x}}(\varepsilon^{\boldsymbol{x}})$ and $P(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}}) \approx (k-1)/L$, so that [88]

$$\hat{p}_{kNN}(\boldsymbol{x}) = \frac{k-1}{L \; V_{\boldsymbol{x}}(\varepsilon^{\boldsymbol{x}})}. \tag{2.16}$$

Note that the k-th nearest neighbor itself is *not* counted as inside the ball (since the ball is open). The k-NN estimator is asymptotically unbiased and consistent [45, 88] provided that k increases as a function of L, with

$$\lim_{L \to \infty} k(L) = \infty$$
$$\lim_{L \to \infty} \frac{k(L)}{L} = 0.$$

$\hat{p}_{kNN}(\boldsymbol{x})$ is also finite sample unbiased under the first order approximation that $p(\boldsymbol{x}) = P(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}})/V_{\boldsymbol{x}}(\varepsilon^{\boldsymbol{x}})$ [45], though this will not hold in general.

The k-NN estimator will apply even when used to estimate $\hat{p}_{kNN}(\boldsymbol{x})$ for $\boldsymbol{x} = \boldsymbol{x}^{(i)}$ with $\boldsymbol{x}^{(i)} \in \mathcal{D}$. In this case, $\boldsymbol{x}^{(i)}$ must be considered as the first nearest neighbor of $\boldsymbol{x}$, at a distance of 0.

The k-NN estimator can be interpreted as a kernel estimator with the choices

$$K(\boldsymbol{a}) = \begin{cases} \frac{1}{2^{D_{\boldsymbol{x}}}}, & \|\boldsymbol{a}\|_\infty < 1 \\ 0, & \text{otherwise} \end{cases}$$

and $h = \varepsilon^{\boldsymbol{x}}$ [91]. However, when multiple estimates of $p(\boldsymbol{x})$ are needed at different $\boldsymbol{x}$, the two methods differ. Where h is typically selected as a constant for all $\boldsymbol{x}$, $\varepsilon^{\boldsymbol{x}}$ depends on the value of $\boldsymbol{x}$ selected. k-NN approaches are favored for their fast empirical convergence rates where samples are dense, however, they are inaccurate for $\boldsymbol{x}$ far from any samples. The fact that $\varepsilon^{\boldsymbol{x}}$ can change means that the ball $\mathscr{B}(\boldsymbol{x}, \varepsilon^{\boldsymbol{x}})$ is made to be as large as needed to include k samples, even when $\boldsymbol{x}$ very far from all samples. In particular, the numerator of (2.16) is never 0, so that $\hat{p}_{kNN}(\boldsymbol{x}) \sim 1/\|\boldsymbol{x}\|$ in areas where there are no samples. This means $\int \hat{p}_{kNN}(\boldsymbol{x})\, d\boldsymbol{x} \neq 1$, whereas $\hat{p}_{kern}(\boldsymbol{x})$ does normalize correctly, with $\int \hat{p}_{kern}(\boldsymbol{x})\, d\boldsymbol{x} = 1$. k-NN estimators are therefore not suitable far from regions where any samples have been observed.

2.6.2 Sample-Based Entropy Estimators

In addition to estimation of probability density, development of query-driven adaptive sampling will require estimation of mutual information. Sample-based mutual information estimators are constructed from sample-based estimators of entropy, so we introduce entropy estimators first. Sample-based estimators of entropy take the form

$$\hat{H}(\boldsymbol{x}) = -\frac{1}{L}\sum_{i=1}^{L} \widehat{\log p}(\boldsymbol{x}^{(i)}), \tag{2.17}$$

where $\widehat{\log p}(\boldsymbol{x})$ is an estimate for $\log p(\boldsymbol{x})$. The choice of notation here draws attention to the fact that $\widehat{\log p}(\boldsymbol{x})$ can be constructed to have lower errors than the 'plug-in estimator', which estimates $\log p(\boldsymbol{x})$ as $\log \hat{p}(\boldsymbol{x})$.

It is known that no finite sample unbiased estimator exists for discrete entropy [103], though it is possible to reduce bias below the level achieved by the plug-in estimator. Grassberger [56] introduced the estimator

$$\widehat{\log p}(\boldsymbol{x}) = G(L_{\boldsymbol{x}}, L) := \psi(L_{\boldsymbol{x}}) + \frac{(-1)^{L_{\boldsymbol{x}}}}{2}\left[\psi\left(\frac{L_{\boldsymbol{x}}+1}{2}\right) - \psi\left(\frac{L_{\boldsymbol{x}}}{2}\right)\right] - \log L \tag{2.18}$$

where $\psi(a)$ is the digamma function. Grassberger showed that bias in the estimator decreases exponentially in expected number of samples of each value of $\boldsymbol{x}$, which is

faster than the plug-in estimator.

In the continuous case, entropy is frequently estimated through the use of the Kozachenko-Leonenko (KL) entropy estimator [74, 89], which builds off of k-NN density estimators. For sample $\boldsymbol{x}^{(i)}$, we let $\varepsilon^{\boldsymbol{x}^{(i)}}$ be the distance from $\boldsymbol{x}^{(i)}$ to its k-th nearest neighbor in $\mathcal{D}$. For consistency with our discussion of k-NN density estimators, we allow the set of neighbors to include $\boldsymbol{x}^{(i)}$.

Our presentation here differs slightly from typical discussions of KL entropy estimators, in which $\log p(\boldsymbol{x}^{(i)})$ is only estimated at a sample $\boldsymbol{x}^{(i)}$ and not for general $\boldsymbol{x} \notin \mathcal{D}$. For this reason, other work uses a set of neighbors of $\boldsymbol{x}^{(i)}$ that do not include $\boldsymbol{x}^{(i)}$ itself. Including $\boldsymbol{x}^{(i)}$ in the set of neighbors makes little difference for numerical computation, but also allows us to generate estimates of log probability at $\boldsymbol{x} \notin \mathcal{D}$.

The KL entropy estimator relies on the identity [89]

$$\mathbb{E}\left[\log P(\boldsymbol{x}^{(i)}, \varepsilon^{\boldsymbol{x}^{(i)}})\right] = \psi(k-1) - \psi(L) \tag{2.19}$$

so that $\psi(k) - \psi(L)$ is an unbiased estimator for the log probability mass within $\mathscr{B}(\boldsymbol{x}^{(i)}, \varepsilon^{\boldsymbol{x}^{(i)}})$. This then leads to the following log probability estimate in the KL estimator

$$\widehat{\log p}(\boldsymbol{x}^{(i)}) = \psi(k-1) - \psi(L) - \log V_{\boldsymbol{x}}(\varepsilon^{\boldsymbol{x}^{(i)}}). \tag{2.20}$$

2.6.3 Sample-Based Mutual Information Estimators

We are now able to estimate mutual information using the entropy estimators in the previous section. Sample-based estimation of mutual information is performed on a set of independent samples $\mathcal{D} = \{(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})\}_{j=1}^{L}$ drawn from $p(\boldsymbol{x}, \boldsymbol{y})$. Mutual information may be estimated using a so-called '3H estimator', which is based on the identity $I(\boldsymbol{x}; \boldsymbol{y}) = H(\boldsymbol{x}) + H(\boldsymbol{y}) - H(\boldsymbol{x}, \boldsymbol{y})$ and uses KL entropy estimators for marginal and joint entropies separately. For each sample $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \in \mathcal{D}$, we compute the distance to the k-th nearest neighbor in the joint space $\mathcal{X} \times \mathcal{Y}$, and the marginal

spaces $\mathcal{X}$ and $\mathcal{Y}$ as $\varepsilon^{\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}}$, $\varepsilon^{\boldsymbol{x}^{(i)}}$, and $\varepsilon^{\boldsymbol{y}^{(i)}}$. Then

$$\hat{I}(\boldsymbol{x};\boldsymbol{y}) = \frac{1}{L}\sum_{i=1}^{L}\Big(\Big[\psi(L) - \psi(k-1) + \log V_{\boldsymbol{x}}(\varepsilon^{\boldsymbol{x}^{(i)}})\Big] + \Big[\psi(L) - \psi(k-1) + \log V_{\boldsymbol{y}}(\varepsilon^{\boldsymbol{y}^{(i)}})\Big] - \Big[\psi(L) - \psi(k-1) + \log V_{\boldsymbol{x},\boldsymbol{y}}(\varepsilon^{\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}})\Big]\Big), \quad (2.21)$$

where $V_{\boldsymbol{x}}$ denotes the volume of the ball only in the marginal space $\mathcal{X}$.

Kraskov et al. [75] recognized that the choice of k for each of the entropy estimators does not have to be equal, and can be computed as the number of samples (including $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$) in a ball of any fixed radius ε. Under a hypothesis that errors in the information estimates were predominantly caused by differences in the volumes of entropy estimators, they elected to fix k in the joint entropy estimator, and compute $\varepsilon^{\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}}$ as the distance to the k-th nearest neighbor in the joint space. They then select $\varepsilon^{\boldsymbol{x}^{(i)}} = \varepsilon^{\boldsymbol{y}^{(i)}} = \varepsilon^{\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}}$, so that under an infinity norm, $V_{\boldsymbol{x}}(\varepsilon^{\boldsymbol{x}^{(i)}})\ V_{\boldsymbol{y}}(\varepsilon^{\boldsymbol{y}^{(i)}}) = V_{\boldsymbol{x},\boldsymbol{y}}(\varepsilon^{\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}})$, and volumes cancel from the estimator.

For each $\boldsymbol{x}^{(i)}$, $k^{\boldsymbol{x}^{(i)}}$ is computed as the number of samples $\boldsymbol{x}^{(j)}$ such that $\left\|\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(i)}\right\| < \varepsilon^{\boldsymbol{x}^{(i)}}$ (including $\boldsymbol{x}^{(j)} = \boldsymbol{x}^{(i)}$), and $k^{\boldsymbol{y}^{(i)}}$ is the number of samples such that $\left\|\boldsymbol{y}^{(j)} - \boldsymbol{y}^{(i)}\right\| < \varepsilon^{\boldsymbol{y}^{(i)}}$ (including $\boldsymbol{y}^{(i)}$). This results in the 'KSG estimator'

$$\hat{I}(\boldsymbol{x};\boldsymbol{y}) = \frac{1}{L}\sum_{i=1}^{L}\left[\psi(L) + \psi(k-1) - \psi(k^{\boldsymbol{x}^{(i)}}) - \psi(k^{\boldsymbol{y}^{(i)}})\right]. \quad (2.22)$$

Despite the form of an empirical expectation of a function applied to each sample, the KSG estimator is *not* an expectation over independent random variables. $k^{\boldsymbol{x}^{(i)}}$ depends on samples $\boldsymbol{x}^{(j)} \neq \boldsymbol{x}^{(i)}$, so that each element of the sum is not independent. Practically, this means that convergence theorems such as the central limit theorem do *not* apply. In particular, the KSG estimator is biased for finite L.

The KSG estimator has been shown to result in more accurate predictions for information than the 3H estimator for a variety of distributions [75], and so we will use the KSG estimator for information estimation in this thesis. We may interpret the information estimator as the empirical average of an estimator for the log probability

ratio $\hat{lr}(\boldsymbol{x}^{(i)}; \boldsymbol{y}^{(i)})$, leading to

$$\hat{lr}(\boldsymbol{x}^{(i)}; \boldsymbol{y}^{(i)}) = \psi(L) + \psi(k-1) - \psi(k^{\boldsymbol{x}^{(i)}}) - \psi(k^{\boldsymbol{y}^{(i)}}). \tag{2.23}$$

In other work, where $k^{\boldsymbol{x}^{(i)}}$ is defined to not count the sample $\boldsymbol{x}^{(i)}$ itself, the KSG estimator is seen with terms of the form $\psi(k^{\boldsymbol{x}^{(i)}} + 1)$, because there are $k^{\boldsymbol{x}^{(i)}} + 1$ total samples within $\mathscr{B}(\boldsymbol{x}^{(i)}, k^{\boldsymbol{x}^{(i)}})$.

2.7 Queries

The specification of a query informs an agent of which locations in an environment are worth exploring. In this section, we introduce queries with three different classes of objectives, and transformations on those queries that we call 'specializations', whose inclusion increases the expressiveness of the query language.

The first class of query objectives are formulated using probabilities $p(\zeta \mid \mathbf{o}_{0:T})$, the second is based on mutual information which is computed using log probabilities $\log p(\zeta \mid \mathbf{o}_{0:T})$, and the third is based on the expected values of the variable of interest ζ that occur. We refer to these as queries with probability objectives, information objectives, and value objectives respectively.

The existing adaptive sampling problems covered in Section 2.2 align directly with two of these classes of query objectives; information maximization is achieved through an information query objective, and maximization of observables is achieved through a value query objective. However, the previously mentioned specializations allow our queries to express a significantly larger set of objectives than existing work, by allowing additional restrictions on the domain of ζ and the subset of outcomes that need to be optimized.

We construct a query in terms of three components. Intuitively, the first component is specification of the variable of interest, which is a quantity to be measured or predicate to be answered using a query function. The second component specifies how the variable of interest should studied through a query objective. The query

objective is one of the previously mentioned value, probability, or information objectives, which state that we are interested in the expected value of the variable of interest, its posterior probability distribution after observations are taken, or mutual information between observations and the variable of interest. The third component, the query threshold, specifies whether the mission should achieve the best possible result in a set number of observations, or collect the fewest possible observations in order to reach a fixed level of belief or precision.

We now formally define a query as an operator on environment variables and model parameters. Following our model specification, a query may request that an agent focuses on environment variables, the spatio-temporal correlations between environment variables, the presence or absence of causal relationships between environment variables, and/or the strength of those relationships.

Definition 1. *A query* $\mathcal{Q} = \langle f_{\mathcal{Q}}, J_{\mathcal{Q}}, \Delta_{\mathcal{Q}} \rangle$ *is a tuple, where* $f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \langle \mathbf{y}, \mathbf{c}, \mathcal{E}, \boldsymbol{\lambda} \rangle)$ *is a query function that returns an output* $\zeta \in \mathcal{Z}$*,* $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ *is a query objective that returns a reward in* $\mathbb{R}$*, and* $\Delta_{\mathcal{Q}}$ *is a query threshold that the query objective must exceed, if specified.*

The following subsections provide more detail on query functions, objectives, and thresholds.

2.7.1 Query Functions

Intuitively, the query function returns an encoding of any variables the user wishes to be the subject of study during a mission. The vector of locations reached by the agent $\boldsymbol{x}_{1:T}$ is included in the query function to allow queries to depend on the path taken by the agent. This is most useful for specifying that a query depends on the locations that were specifically observed, like when an agent needs to produce proof of a specific phenomenon.

Since query functions can be any function a user can construct, they can be used for diverse classes of problems, including continuous quantitative problems, logic problems, and classification problems. We highlight these below with examples.

Continuous Quantitative Problems

Simple queries may focus on certain environment variables or parameters, in which case, they may be returned directly by the query function. For example, a query focusing on the length scale of spatial correlations $l_m \in \mathbf{c}_m$ may be defined simply as

$$f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \langle \mathbf{y}, \mathbf{c}, \mathcal{E}, \boldsymbol{\lambda} \rangle) = l_m.$$

Alternatively, a query focusing on the strength of the relationship between $\mathbf{y}_m$ and $\mathbf{y}_n$ may be defined as

$$f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \langle \mathbf{y}, \mathbf{c}, \mathcal{E}, \boldsymbol{\lambda} \rangle) = \boldsymbol{\lambda}_{m,n}.$$

It is possible to return any function of the environment variables and model parameters from the query function, and this may be used to define derived outputs, or conditions on observations. For example, a query focusing on the maximum value of attribute $\mathbf{y}_m$ across the domain may be defined as

$$f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \langle \mathbf{y}, \mathbf{c}, \mathcal{E}, \boldsymbol{\lambda} \rangle) = \max(\mathbf{y}_m).$$

Logical Problems

A query function may also return the answer to a boolean question, with 1 indicating truth and 0 indicating falsehood. This means queries may focus on the answers to questions specified by formulations of logic.

For example, a query focusing on the boolean question of whether $\mathbf{y}_m$ is directly caused by $\mathbf{y}_n$ and $\mathbf{y}_l$ may be defined as

$$f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \langle \mathbf{y}, \mathbf{c}, \mathcal{E}, \boldsymbol{\lambda} \rangle) = \mathbb{1}\left\{(n \to m) \in \mathcal{E} \ \wedge \ (l \to m) \in \mathcal{E}\right\},$$

where $\mathbb{1}\{a\}$ is a function that returns 1 if the local expression a is true, and 0 otherwise.

A query seeking to observe, at any location, evidence of a mound with active

seepage may define a query function as

$$f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \langle \mathbf{y}, \mathbf{c}, \mathcal{E}, \boldsymbol{\lambda} \rangle) = \bigvee_{t=1}^{T} \mathbb{1}\left\{ \mathbf{y}_1(\boldsymbol{x}_t) = 1 \wedge \mathbf{y}_2(\boldsymbol{x}_t = 1) \right\},$$

if $\mathbf{y}_1$ and $\mathbf{y}_2$ are the presence of mounds and active seepage respectively.

Classification Problems

A query may also return a classification that is derived from the attributes. It is not necessary for the classifications to be cast to a numeric number. These types of query functions support labeling areas in the environment as part of distinct categories.

For example, we could define a coral reef as 'healthy' if less than 2% of coral are bleached, 'threatened' if 2-10% of coral are bleached, and 'endangered' if more than 10% of coral are bleached. If $\mathbf{y}_1(\boldsymbol{x})$ is an attribute that is 1 if coral exists at location $\boldsymbol{x}$ and 0 otherwise, and $\mathbf{y}_2(\boldsymbol{x})$ is an attribute that is 1 if coral at a location $\boldsymbol{x}$ is bleached and 0 otherwise, we may define a query function as

$$f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \langle \mathbf{y}, \mathbf{c}, \mathcal{E}, \boldsymbol{\lambda} \rangle) = \begin{cases} \textit{healthy}, & \frac{\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}\{\mathbf{y}_1(\boldsymbol{x})=1 \wedge \mathbf{y}_2(\boldsymbol{x})\}}{\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}\{\mathbf{y}_1(\boldsymbol{x})=1\}} < 0.02 \\ \textit{threatened}, & 0.02 \leq \frac{\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}\{\mathbf{y}_1(\boldsymbol{x})=1 \wedge \mathbf{y}_2(\boldsymbol{x})\}}{\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}\{\mathbf{y}_1(\boldsymbol{x})=1\}} \leq 0.1 \\ \textit{endangered}, & \frac{\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}\{\mathbf{y}_1(\boldsymbol{x})=1 \wedge \mathbf{y}_2(\boldsymbol{x})\}}{\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}\{\mathbf{y}_1(\boldsymbol{x})=1\}} > 0.1. \end{cases}$$

There is no requirement that $f_{\mathcal{Q}}$ is a simple function of its parameters, as in these examples. The output may be given by any computational routine. For example, the query function may solve an optimization problem that is formulated from the observations and parameters. This allows us to express important queries, such as what an upper bound on ocean depth for a region. In our experiments, we use query-directed adaptive sampling to find unblocked escape routes through a road network by defining $f_{\mathcal{Q}}$ to be the output of a graph search routine.

There is a technical issue in that the query function may refer parameters for edges in a DAG structural model of the environment that do not exist in every instance of the model. For example, the set of edges $\mathcal{E}$ is itself a random variable, so that the

edge parameters $\boldsymbol{\lambda}_{m,n}$ do not exist unless $(n \to m) \in \mathcal{E}$. Dependent on the goal, the user may either check for the existence of $(n \to m)$ in $f_{\mathcal{Q}}$ and return a specified value when it does not exist, or specify that the model must include an edge $(n \to m)$.

2.7.2 Overview of Query Objectives

In this and the next two sections, we will introduce query objectives. Intuitively, the query objective describes how a measure of reward should be computed from the output of the query function.

We desire for our query objectives to cover a large portion of the problems expressible by the current literature, including information maximization and observable maximization problems. For this reason, a query objective includes a basic objective, which specifies whether the mission is concerned with optimizing expectations of variables, posterior probabilities, or mutual information. However, we find that this alone is not sufficient to express all the objectives in which we are interested. For this reason, we also introduce posterior and prior specializations, which respectively constrain the values of ζ and $\mathbf{o}_{1:T}$ that are of interest in the problem. A query objective may include zero or one posterior specializations and zero or one prior specializations.

Mathematically, a query objective is always a function $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ that depends on the distributions of the random variables ζ and $\mathbf{o}_{1:T}$, and the inclusion of specializations changes the form of the function. In practice, the basic objective is one of three options, a posterior specialization is fully specified by a set $\mathcal{V}$, and a prior specialization is specified by a number $0 \leq \delta_{prior} \leq 1$. Therefore, in order for a user to specify a query objective, they may provide a tuple

$$\langle J_{basic}, \mathcal{V}, \delta_{prior} \rangle ,$$

where $J_{basic} \in \{value, probability, information\}$, and $\mathcal{V}$ and δ_{prior} can be optionally excluded to indicate that no posterior or prior specializations are necessary.

2.7.3 Basic Query Objectives

We first describe query objectives in their simplest forms, without introducing modifiers that restrict the objectives to specific values. Unlike changes in the query function, changes in the query objective impact the quantities estimated during planning, and so change how the adaptive sampling algorithm operates. The basic choice of query objective is one of three options:

Maximization of Value: This objective indicates that the agent must select observations that maximize the expectation of the variable of interest,

$$J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}\left[\zeta \mid \mathbf{o}_0\right]. \tag{2.24}$$

A value objective should be used when a mission aims to maximize observations of an environmental variable. This objective depends on the agent's actions when ζ depends on $\boldsymbol{x}_{1:T}$. This is most useful for attempting to locate specific phenomena, for example maximizing the number of hydrocarbon seeps observed, or the number of locations where conditions on local environment variables are met. Use of this objective requires that the output of the objective function is scalar and numeric.

Maximization of Probability: This objective indicates that the agent must select observations that maximize the expected posterior probability of the variable of interest ζ,

$$J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}\left[p(\zeta \mid \mathbf{o}_{0:T}) \mid \mathbf{o}_0\right]. \tag{2.25}$$

This objective gives an interpretable measure of the posterior belief in the variable of interest ζ after the observations have been taken. Expected belief is not a rigorous measure of certainty like Shannon entropy, but it is easier to interpret than information and does not depend upon prior belief $p(\zeta \mid \mathbf{o}_0)$. Probability objectives are most useful when used in combination with specializations to maximize the probability of a particular outcome. This should be used when the results from a query will be used to make a single informed estimate of the variable of interest. An example is

maximizing the posterior belief in the number of seeps in the environment. A probability objective would be the correct choice when a single number of seeps in the environment must be estimated, with a penalty when that selection is wrong.

Maximization of Information: This objective indicates that the agent must select observations to maximize mutual information between observations and the variable of interest,

$$J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = I(\zeta; \mathbf{o}_{1:T} | \mathbf{o}_0). \tag{2.26}$$

Maximization of mutual information is equivalent to minimizing posterior Shannon entropy of ζ conditioned on $\mathbf{o}_{1:T}$. This means that an information objective should be used when a mission aims to reduce uncertainty in ζ, as measured by entropy, as much as possible. For example, an information objective could be used to generate the most precise estimates for the number of seeps in the environment. An information objective would be the correct choice when the full posterior distribution of possibilities would be considered, without making a single final choice or estimation.

All three classes of objectives maximize some quantity. More complex queries, like determining whether a quantity falls within a certain range, are expressible with specializations and sufficiency, which we cover in the following subsections.

It will be convenient in later definitions and proofs to refer to all query objectives with a single notation, so we introduce two functions $f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ and $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$ for this purpose. f is the quantity within expectations in each query objective, while g includes an expectation over ζ. f and g allow us to express any query objective as

$$\begin{aligned} J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) &= \mathbb{E}\left[f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right] \\ &= \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[\mathbb{E}_{\zeta|\mathbf{o}_{0:T}}\left[f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_0\right] \\ &= \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right]. \end{aligned} \tag{2.27}$$

The basic query objectives may be expressed using definitions of f given in Table 2.1, and with $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}_{\zeta|\mathbf{o}_{0:T}}\left[f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right]$. As such, further modifications to queries will be discussed in this form, with the understanding that they are applicable

Objective	$f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$	$\mathcal{V}(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$
Value	ζ	$\mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ or $\mathcal{V}_{const}$
Probability	$p(\zeta \mid \mathbf{o}_{0:T})$	$\mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$
Information	$lr(\zeta; \mathbf{o}_{1:T} \mid \mathbf{o}_0)$	None

Table 2.1: Summary of objectives and posterior specializations available in query-driven adaptive sampling.

to each of the three basic objective forms above.

To illustrate f and g through example, let ζ be the the number of seeps in the environment and consider a probability objective. Then

$$f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = p(\zeta \mid \mathbf{o}_{0:T})$$

is the probability of a specific value of the number of seeps conditioned on specific observations $\mathbf{o}_{0:T}$.

$$g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}_{\zeta|\mathbf{o}_{0:T}}\left[p(\zeta \mid \mathbf{o}_{0:T}) \mid \mathbf{o}_{0:T}\right]$$

is the expected posterior probability over all possible numbers of seeps, conditioned on specific observations $\mathbf{o}_{0:T}$. Finally,

$$J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[\mathbb{E}_{\zeta|\mathbf{o}_{0:T}}\left[p(\zeta \mid \mathbf{o}_{0:T}) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_0\right]$$

is the expected posterior probability probability over all possible numbers of seeps, conditioned on all possible values of the observations $\mathbf{o}_{1:T}$, and the specific observations $\mathbf{o}_0$.

Later in this chapter, when we discuss conditions required for queries to be well-formed and when we discuss online planning, we will need to understand how an objective changes when conditioning on a larger set of observation than before, such as when conditioned on$\mathbf{o}_{0:t+1}$ after conditioning on $\mathbf{o}_{0:t}$. For now, we point out that for value and probability objectives, $f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = f(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ and $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) = g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ for $1 \leq t \leq T$, while this is not true for information objectives. We will use this fact in our proofs later.

2.7.4 Specialization in Query Objectives

In this section, we add complexity to query objectives to express more interesting problems by introducing specializations. In their basic forms, the query objectives above significantly overlap with approaches to adaptive sampling elsewhere in the literature. But these objectives alone are not sufficient to model some of the more complex queries we would like to consider. Examples of these more complex queries include "Maximize the expected number of seeps found on the best 90% of mission outcomes", and "Maximize the expected posterior mode of the number of seeps in the environment". In these examples, focusing on the best 90% of mission outcomes and focusing on the posterior mode of a distribution are modifiers to a basic query objective that existing adaptive sampling approaches could not model.

To allow these queries to be modeled and answered, we now show how our query objectives may be specialized to only consider certain values of the variable of interest, such as the posterior mode, or specific observations, including the top 90% of mission outcomes. Here, we introduce two kinds of specializations, which we name prior and posterior specializations. A prior specialization constrains the prior values of observations that are considered in the objective, and would allow a query to focus on the top 90% of mission outcomes. Meanwhile, a posterior specialization constrains the posterior values of the variable of interest, conditioned on $\mathbf{o}_{0:T}$, that are considered in the objective, and would allow a query to optimize a posterior mode. Including these specializations in our query definition allows for adaptive sampling objectives significantly beyond the state of the art.

To intuitively describe the queries that specializations allow, we illustrate them in terms of our motivating scenario, where a vehicle explores the ocean floor, and observes sites of hydrocarbon seepage, given as attribute $\mathbf{y}_1$. We consider two queries of interest. The first is the number of seeps at the locations that the vehicle has visited,

$$\zeta_1 = \sum_{t=1}^{T} \mathbb{1}\left\{\mathbf{y}_1(\boldsymbol{x}_t) = 1\right\},$$

while the second is the total number of seeps in the environment,

$$\zeta_2 = \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}\left\{\mathbf{y}_1(\boldsymbol{x}) = 1\right\}.$$

ζ_1 describes only what is seen on the mission, while ζ_2 describes the true state of the world. Based on camera imagery, the vehicle provides observations of whether a seep is present at the locations that it visits. However, the images may be misinterpreted, so the observations may be considered noisy observations of the truth.

Types of Specializations

To motivate and better distinguish between prior and posterior specializations, let us return to the previous statement that all basic queries can be formulated as expected values. The expectations may be expanded as an outer expectation over $\mathbf{o}_{1:T} \mid \mathbf{o}_0$, the observations that could be received in the mission, and an inner expectation over $\zeta \mid \mathbf{o}_{0:T}$, the variable of interest conditioned on the observations taken during the mission,

$$\mathbb{E}\left[f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right] = \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[\mathbb{E}_{\zeta|\mathbf{o}_{0:T}}\left[f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_0\right]. \tag{2.28}$$

The outer expectation above considers all possible values of observations during the mission conditioned on the initial observations, $\mathbf{o}_{1:T} \mid \mathbf{o}_0$, while the inner expectation considers all possible values of the variable of interest conditioned on the observations, $\zeta \mid \mathbf{o}_{0:T}$. The goal is to select observations that make the expectation as large as possible. Specialization adds in the idea that not all values of ζ and $\mathbf{o}_{1:T}$ are equally important for the mission. Considering the nested expectations explicitly reveals that these concepts can be applied to both the distributions $p(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$ and $p(\zeta \mid \mathbf{o}_{0:T})$. Posterior specializations limit the values of ζ considered in $p(\zeta \mid \mathbf{o}_{0:T})$, while prior specializations limit the values of $\mathbf{o}_{1:T}$ considered in $p(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$.

Posterior Specialization

In the query "Maximize the expected posterior mode of the number of seeps in the environment", we are not concerned with the probabilities of all possible numbers of seeps on the environment, only the mode. This idea is encompassed in the concept of *posterior* specialization, which focuses on the posterior probability that ζ falls within a specific domain of interest. In the seep count example just given, we are only interested in $P\left[\zeta_2 = \arg\max_{\zeta_2'} \; p(\zeta_2' \mid \mathbf{o}_{0:T}) \mid \mathbf{o}_{0:T}\right]$, and not in the distribution over the values of ζ_2 that are not the mode.

A posterior specialization allows a user to specify a query such as "Maximize the expected posterior probability that at least three seeps are visited". The number of seeps visited follows a distribution $p(\zeta_1 \mid \mathbf{o}_{0:T})$, so there is a probability $P\left[\zeta_1 \geq 3 \mid \mathbf{o}_{0:T}\right]$ that three or more seeps have been visited for each $\mathbf{o}_{1:T}$ that could be observed. This is the probability that ζ_1 is a member of a fixed subset of its domain, namely the set of numbers greater than 3. The objective then maximizes the expectation of that probability over different possible observations. In this case, that objective would be to find the policy π that satisfies

$$\max_{\pi} \; \mathbb{E}_{\mathbf{o}_{1:T} \mid \mathbf{o}_0}\left[P\left[\zeta_1 \geq 3 \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_0\right]. \tag{2.29}$$

Alternatively, a user might ask "Maximize the expected posterior mode of the number of seeps in the environment". The number of seeps in the environment follows a distribution $p(\zeta_2 \mid \mathbf{o}_{0:T})$, and the posterior mode is $\max_{\zeta_2} p(\zeta_2 \mid \mathbf{o}_{0:T})$. This is also the probability that ζ_2 falls in a set; the set consisting of the the values of ζ_2 with greatest probability. In contrast to the previous example, the set is now not fixed, as different values of ζ_2 may have the highest probability when conditioned on different observations $\mathbf{o}_{1:T}$. This objective maximizes the expectation of that mode

over different possible observations, so the objective would be

$$\max_{\pi} \; \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0} \left[\max p(\zeta_2 \mid \mathbf{o}_{0:T}) \,|\, \mathbf{o}_0 \right] = \max_{\pi} \; \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0} \left[P \left[\zeta_2 = \arg\max_{\zeta_2'} \; p(\zeta_2' \mid \mathbf{o}_{0:T}) \,\middle|\, \mathbf{o}_{0:T} \right] \middle|\, \mathbf{o}_0 \right]. \quad (2.30)$$

Both the example queries above may be expressed through application of a posterior specialization, which we now define. A posterior specialization is used to transform the inner expectation over $p(\zeta \mid \mathbf{o}_{0:T})$ in an objective into a probability of membership in a set $\mathcal{V}(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$, as

$$\mathbb{E}_{\zeta|\mathbf{o}_{0:T}} \left[f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T} \right] \rightarrow P \left[\zeta \in \mathcal{V}(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T} \right]. \quad (2.31)$$

This transformation results in a different kind of objective, one that is focused on probability of membership in a set $\mathcal{V}(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ that potentially depends on the class of query objective and observations. For example, the set of values of ζ with highest probability conditioned on $\mathbf{o}_{0:T}$ depends upon the observations taken and the fact that the query objective focuses on probabilities. Posterior specializations are introduced in this way because many queries of interest are concerned with the probability that a variable of interest is within some desired range.

The transformation of a posterior specialization results in a final objective of

$$\max_{\pi} \; \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0} \left[P \left[\zeta \in \mathcal{V}(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T} \right] \right], \quad (2.32)$$

whereas previously it was expressed as (2.28).

It should be noted that this objective is equivalent to

$$\max_{\pi} \; P \left[\zeta \in \mathcal{V}(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0 \right], \quad (2.33)$$

but we emphasize the potential dependence of $\mathcal{V}$ on $\mathbf{o}_{0:T}$, meaning that the value of the objective is not independent of the observations taken.

Only certain forms of $\mathcal{V}(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ are permitted in order to avoid unscientific behavior by the agent. We provide a technical discussion of the rationale behind these restrictions, and the aberrant behavior that we avoid, in Section 2.8.

For value objectives, $\mathcal{V}(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ may be specified by the user to be any constant subset of the domain of ζ, which we denote by $\mathcal{V}_{const}$. This is useful when only certain query outputs are of interest. The previous example to "Maximize the expected posterior probability that at least three seeps are visited" was an example of a constant set, with $\mathcal{V}_{const} = \{i \mid i \geq 3\}$.

Additionally, extreme values of $f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ may be of interest, without the user necessarily knowing what they are for each possible value of $\mathbf{o}_{0:T}$. Examples include maximizing the largest possible query output or the probability of the mode. Queries may focus on larger sets than the single top value, particularly when ζ is continuous. To express these queries, we also allow $\mathcal{V}(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ to be chosen to be the set of elements of $\mathcal{Z}$ with positive probability and maximum cardinality S that maximize $f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$, which we denote as $\mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$. This choice may be made for value and probability objectives.

To formally define $\mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$, let $\mathcal{Z}^+(\mathbf{o}_{0:T})$ be the set of query function outputs with positive probability conditioned on $\mathbf{o}_{0:T}$,

$$\mathcal{Z}^+(\mathbf{o}_{0:T}) := \{\zeta \in \mathcal{Z} \mid p(\zeta \mid \mathbf{o}_{0:T}) > 0\}. \tag{2.34}$$

Then we define

$$\mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) := \underset{\mathcal{V} \subseteq \mathcal{Z}^+(\mathbf{o}_{0:T}), |\mathcal{V}| \leq S}{\arg\max} \sum_{\zeta \in \mathcal{V}} f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0). \tag{2.35}$$

For value objectives, $\mathcal{V}^*_S$ is the set of maximum values of ζ with positive probability, while for probability objectives, it is the set of values with the maximum probability. In the probability objective case, $\mathcal{V}^*_S$ may not be uniquely defined. But since we are only concerned with $P[\zeta \in \mathcal{V}(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}]$, which is equal for all maxima of (2.35), the choice between maximum values can be made arbitrarily.

Information objectives do not support posterior specialization. The options for posterior specializations are summarized in the second column of Table 2.1.

For the examples given above, the objective specified in (2.29) is a value objective using $\mathcal{V}_{const} = [3, \infty)$, and (2.30) is a probability objective using $\mathcal{V}_1^*(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$.

Prior Specialization

In the query "Maximize the expected number of seeps found on the best 90% of mission outcomes", we are not concerned with an expectation over all values of $\mathbf{o}_{1:T}$, but only the 90% of $\mathbf{o}_{1:T}$ that result in the highest number of seeps found. This idea is encompassed in *prior* specialization, which constrains values of observations in the distribution $p(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$. In the seep observation example just given, we are only interested in values of $\mathbf{o}_{1:T}$ that occur with 90% total probability.

Prior specializations may be desired for a number of reasons. There may be certain values of $\mathbf{o}_{1:T}$ which are unlikely, but lead to imprecise posterior estimates of $\zeta \mid \mathbf{o}_{0:T}$. For example, there may be 10% probability that noisy sensors indicate that there are no seeps in the environment when they have, in fact, been observed. If a user is able to characterize the noise associated with their sensors, then they may use a prior specialization to exclude those observations from computation of reward. Another potential use case is when unlikely outcomes lead to very low reward, which can cause a plan to favor more conservative missions that avoid these low reward observations. Depending on the mission, it can be acceptable to remove these low probability, worst case outcomes from the objective.

A prior specialization allows a user to specify a query such as "Maximize the expected number of seeps visited on the best 90% of mission outcomes". In this case, that objective would be

$$\max_{\pi, \mathcal{U}} \mathbb{E}_{\mathbf{o}_{1:T} \mid \mathbf{o}_0} \left[\mathbb{E}_{\zeta \mid \mathbf{o}_{0:T}} \left[\zeta_1 \mid \mathbf{o}_{0:T} \right] \mid \mathbf{o}_0, \mathbf{o}_{1:T} \in \mathcal{U} \right] \text{ s.t. } P\left[\mathbf{o}_{1:T} \in \mathcal{U} \mid \mathbf{o}_0 \right] \geq 0.9. \tag{2.36}$$

A prior specialization can also be combined with a posterior specialization. For example, a user may ask to "Maximize the expected posterior mode of the number of

seeps in the environment on the best 95% of mission outcomes". In this case, there is a 5% chance that the policy could lead to outcomes where the number of seeps is not well known, but those are ignored as low probability outcomes, leading to an objective of

$$\max_{\pi,\mathcal{U}} \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[P\left[\zeta_2 = \arg\max_{\zeta_2'} p(\zeta_2' \mid \mathbf{o}_{1:T}) \,\middle|\, \mathbf{o}_{0:T}\right] \,\middle|\, \mathbf{o}_0, \mathbf{o}_{1:T} \in \mathcal{U}\right]$$
$$\text{s.t. } P\left[\mathbf{o}_{1:T} \in \mathcal{U} \mid \mathbf{o}_0\right] \geq 0.95. \quad (2.37)$$

As in the two examples above, a prior specialization restricts the values of $\mathbf{o}_{1:T}$ considered in the outer expectation $\mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right]$. $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$, may be drawn from any objective with or without posterior specialization. The expectation is conditioned on a set of observation outcomes $\mathcal{U}(\mathbf{o}_{1:T})$ that maximize the conditional expectation and occur with at least probability δ_{prior}, resulting in the following transformation

$$\mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right] \rightarrow \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0, \mathbf{o}_{1:T} \in \mathcal{U}(\mathbf{o}_{1:T})\right] \quad (2.38)$$

where

$$\mathcal{U}(\mathbf{o}_{1:T}) := \arg\max_{\mathcal{U}\subseteq\mathcal{O}, P[\mathbf{o}_{1:T}\in\mathcal{U}|\mathbf{o}_0]\geq\delta_{prior}} \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0, \mathbf{o}_{1:T} \in \mathcal{U}\right] \quad (2.39)$$

The user only needs to specify the constant δ_{prior}, while the adaptive sampling algorithm determines the best set $\mathcal{U}$ that optimizes the expectation.

For the examples given above, (2.36) is a value objective with no posterior specialization and a prior specialization with $\delta_{prior} = 0.9$. (2.37) is a probability objective with a posterior specialization to $\mathcal{V}_1^*(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ and a prior specialization with $\delta_{prior} = 0.95$.

2.7.5 Query Thresholds

Finally, we wish to be able to consider queries that do not run for a set number of observations, but instead specify a termination condition that must be satisfied. An example is "Minimize the expected number of observations so that the probability that 3 or more seeps are found exceeds 70%". This idea is encompassed in the idea of sufficiency, which specifies that reaching a threshold is sufficient to consider the mission complete. Sufficiency transforms the objective from a maximization over a finite time horizon to a minimization of observations.

A sufficient condition specifies that a mission is only required to be run until $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$ within an objective, evaluated for the observations $\mathbf{o}_{1:T}$ that are seen, exceeds a threshold $\Delta_{\mathcal{Q}}$. In the previous example to "Minimize the expected number of observations so that the posterior probability that at least three seeps are visited is at least 70%", that objective would be

$$\min_{\pi} \ \mathbb{E}[T] \ \text{ s.t. } \ P[\zeta_1 \geq 3 \mid \mathbf{o}_{0:T}] \geq 0.7 \quad \forall \mathbf{o}_{1:T}. \tag{2.40}$$

Here it is understood that T is now variable, depending on an agent's choice to continue after each observation.

Alternatively, combining with prior and posterior specializations, a user may ask to "Minimize the expected number of observations so that the posterior mode of the number of seeps in the environment on the best 95% of mission outcomes is at least 80%". In this case, the objective would be

$$\begin{aligned} \min_{\pi,\mathcal{U}} \quad & \mathbb{E}[T] \\ \text{s.t.} \quad & P\left[\zeta_2 = \arg\max_{\zeta_2'} \ p(\zeta_2' \mid \mathbf{o}_{1:T}) \,\middle|\, \mathbf{o}_{0:T}\right] \geq 0.8 \quad \forall \mathbf{o}_{1:T} \in \mathcal{U} \\ & P[\mathbf{o}_{1:T} \in \mathcal{U} \mid \mathbf{o}_0] \geq 0.95. \end{aligned} \tag{2.41}$$

As in the examples above, introducing sufficiency modifies the objective to be the minimum number of observations required for the original object to exceed some

threshold $\Delta_{\mathcal{Q}}$, as

$$\max_{\pi} \; \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0} \left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0 \right] \rightarrow \min_{\pi} \; \mathbb{E}\left[T\right] \text{ s.t. } g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \geq \Delta_{\mathcal{Q}} \quad \forall \mathbf{o}_{1:T} \in \mathcal{U}. \tag{2.42}$$

Without an explicit prior specialization, $\mathcal{U}$ contains all possible observations. In this thesis, we only account for minimization of number of observations. Extensions to minimize other quantities such as fuel usage may be considered as part of future work.

Sufficiency captures the idea that certain missions could be run effectively indefinitely, with diminishing returns. For example, an underwater vehicle could explore an area of several square kilometers for months if given the capability to recharge. However, eventually the data gathered answers the question that prompted the mission with a sufficient level of accuracy, and the resources can instead be used for an alternative purpose. Encoding a level of sufficiency in the query provides a way to evaluate when the mission should be considered complete.

The option to include sufficiency in a query further justifies why prior specializations are necessary. For most environment models, the objective in (2.40) may actually be ill-formed, because there is a non-zero probability that the environment may not contain three or more seeps to find, resulting in $\mathbb{E}[T] = \infty$. This expectation will be infinite so long as there is any positive probability that there is not three or more seeps to find.

A suitably chosen prior specialization allows those possibilities to be excluded from the objective, so that $\mathbb{E}\left[T\right]$ is finite. For example, a user selects a prior confidence δ_{prior} that is below the probability that there are three or more seeps to find. When planning, the observations $\mathbf{o}_{1:T}$ that provide evidence that there are less than three seeps to find will be excluded from $\mathcal{U}$, so the query threshold will not have to be satisfied those outcomes. When those outcomes are detected, the mission ends immediately, and for the remaining outcomes, the number of measurements required to reach the query threshold is finite. As a result, $\mathbb{E}\left[T\right]$ is finite.

This issue of infinite numbers of observations needed to reach certain conditions is the exact issue discussed by Jiang et al. [65]. They resolve the issue by including

a maximum possible value of T, corresponding to each $\boldsymbol{x} \in \mathcal{X}$ being sampled once. But when there is observation noise, repeated observations of the same site may in fact be appropriate, and inclusion of prior specializations can ensure the problem is well-defined even in these cases.

2.8 Ensuring Queries are Well-Formed

Our query formulations are broad and encompass many known formulations of adaptive sampling. But it is of course possible to define objectives that cannot be expressed in our query language. To justify the restrictions in our query language, we now show that some natural objectives can lead to counter-intuitive behavior and bias when used as objectives for adaptive sampling, and we argue that this makes them ill-formed for use as queries that guide missions. We discuss these limitations formally, and prove that our queries do not result in the same behavior.

When a query is ill-formed, the resulting bias in observations means that the results will not necessarily reflect the reality of the environment. For example, if a query asks an agent to prove a certain hypothesis, an agent may ignore all observations that do not support that hypothesis. It will eventually gather a large number of observations that support the hypothesis and conclude that it is true, even when most evidence in the environment points to the hypothesis being false. This leads to misleading conclusions. For this reason, we recommend against planning with ill-formed queries, and we argue that the ill-formed query is rarely the true intention of the user. Users should instead use a well-formed query that is closer to their true intention. For example, a query posed to determine with maximum accuracy whether the hypothesis is true or false will consider both possibilities and will not bias the conclusions in the data in the same way.

Additional queries certainly exist that are not expressible in our query language but are still well-formed. Future work may focus on adding support for these queries, but we suggest that they should be carefully analyzed to prove they are well-formed first.

2.8.1 An Example of an Ill-Formed Query

First, let us consider an example of an objective that is useful, but results in undesirable behavior when used in adaptive sampling. A user may desire to reach a specified level of confidence in the answer to a query after observations have been taken, and to have that confidence level reached for as many outcomes of the mission as possible. In a seep environment, this could take the form of a query to "Maximize the probability that the posterior mode of the number of seeps in the environment exceeds 75%",

$$\max_{\pi} \; P\left[\left(\max_{\zeta_2} \; p(\zeta_2 \mid \mathbf{o}_{0:T})\right) \geq 0.75 \,\middle|\, \mathbf{o}_0\right]. \tag{2.43}$$

For any policy, (2.43) can certainly be evaluated. However, we now show that using it as an objective may cause an agent to deliberately avoid improving its model of the environment, even if additional observations could be gathered for no cost. Example 1 gives an explicit numerical example of how this can occur in this query.

Example 1. *Assume that no prior observations have been taken, and consider* $\zeta_2 \in \{0, 1\}$*, so that there are either 0 or 1 seeps in the environment. Consider a model where* $P[\zeta_2 = 0] = 0.8$ *before any observations. Assume the agent is given the choice of whether to take an observation* $\mathbf{o}$ *at its current location. The observation satisfies* $\mathbf{o} \in \{0, 1\}$*, with*

$$P[\mathbf{o} = 0 \mid \zeta_2 = 0] = 0.9$$
$$P[\mathbf{o} = 0 \mid \zeta_2 = 1] = 0.2$$

so we expect observation of $\mathbf{o}$ *to be useful in determining the true value of* ζ_2*. Application of Bayes' theorem confirms this intuition, and yields*

$$P[\zeta_2 = 0 \mid \mathbf{o} = 0] \approx 0.947$$
$$P[\zeta_2 = 0 \mid \mathbf{o} = 1] \approx 0.333,$$

with $P[\mathbf{o} = 0] = 0.76$*. Depending on the value of* $\mathbf{o}$ *observed, uncertainty in the value*

of ζ may rise or fall, but the likelihood that it rises is low.

For the objective given in (2.43), choosing not to receive the observation $\mathbf{o}$ will result in a larger objective value than receiving it. This occurs because reducing the uncertainty in ζ_2 does not improve the objective, but increasing the uncertainty does reduce it.

If the observation is never taken, or the result is discarded without looking at it, then the objective value is based on the prior belief $p(\zeta_2)$, without any observations. Following (2.43), the objective value is evaluated to be

$$P\left[\left(\max_{\zeta_2} \; p(\zeta_2)\right) \geq 0.75\right] = 1.$$

On the other hand, if the observation is taken, the objective value is based on its posterior belief $p(\zeta_2 \mid \mathbf{o})$, which differs depending on the value of $\mathbf{o}$ observed. The confidence threshold of 0.75 is only satisfied when $\mathbf{o} = 0$, and so the objective value is

$$P\left[\left(\max_{\zeta_2} \; p(\zeta_2 \mid \mathbf{o})\right) \geq 0.75\right] = P\left[\mathbf{o} = 0\right] = 0.76.$$

The agent therefore maximizes the objective by avoiding observation of $\mathbf{o}$, because it can result in undesirable outcomes, even if it is already at an appropriate location to take the measurement with no cost.

In this example, the agent avoids the observation because its former belief about the environment is favorable compared to possible posterior beliefs. The agent will only select observations that do not reduce the posterior mode below 0.75, biasing the data gathering process in order to reinforce the current belief state.

When an agent avoids data and deliberately biases its understanding of the environment, it acts in an unscientific manner. Because of the bias, the results may also not be reflective of the true probabilities of query satisfaction. We view this as undesirable behavior, and for this reason label (2.43) as an ill-formed query.

2.8.2 Monotonically Nondecreasing Queries

In the previous example, the agent avoided gathering the observation $\mathbf{o}$ because the objective value decreased with the addition of $\mathbf{o}$. More generally, an agent will not avoid gathering data if it can be proven that the objective value will not decrease as the result of an observation that can be gathered without cost. To avoid ignoring observations, we assert that all query objectives must be *monotonically nondecreasing*, defined below. When an objective is not monotonically nondecreasing, the value of the objective can decrease with additional observations, causing an agent to potentially discard or refuse to gather data.

Definition 2. *An objective $J_{\mathcal{Q}}$ is said be monotonically nondecreasing if, for every observation $\mathbf{o}_T$ that does not affect the true value of ζ, that $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \geq J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0)$.*

Note that the definition of monotonically nondecreasing objectives does not consider the case where observation $\mathbf{o}_T$ decreases the objective because the true value of ζ is changed. This can occur for choices of query functions that depend directly on the observation locations $\boldsymbol{x}_{1:T}$, such as a variable of interest equal to the minimum temperature observed on the mission. In this case, reward for biasing the observations has been built in to the query function, and does not indicate that the query objective is unsuitable.

We now show that the query objectives and specializations that we have defined are monotonically nondecreasing, and so are appropriate for use as objectives in adaptive sampling missions. Theorems 1 through 3 show that value, probability, and information objectives without specializations are all monotonically nondecreasing.

Theorem 1. *A value objective is monotonically nondecreasing.*

Proof. Without posterior specialization, the value objective $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}[\zeta \mid \mathbf{o}_0]$ does not directly depend on the observations. If the value of ζ is unchanged by $\mathbf{o}_T$, then $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) = J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$. □

Theorem 2. *A probability objective is monotonically nondecreasing.*

Proof. The result follows from application of Sedrakyan's inequality [122], which states that for real $a_1, \ldots, a_n$ and positive $b_1, \ldots, b_n$,

$$\sum_{i=1}^{n} \frac{a_i^2}{b_i} \geq \frac{\left(\sum_{i=1}^{n} a_i\right)^2}{\sum_{i=1}^{n} b_i}.$$

For fixed $\mathbf{o}_{0:T-1}$, and observations $\mathbf{o}_T$ with non-zero probability, select $a_i = p(\zeta \mid \mathbf{o}_{0:T})\ p(\mathbf{o}_T \mid \mathbf{o}_{0:T-1})$ and $b_i = p(\mathbf{o}_T \mid \mathbf{o}_{0:T-1})$. Then when the addition of $\mathbf{o}_T$ does not change the true value of ζ, we have

$$\begin{aligned}
J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) &= \mathbb{E}\left[p(\zeta \mid \mathbf{o}_{0:T})\right] \\
&= \sum_{\zeta} \sum_{\mathbf{o}_{0:T-1}} p(\mathbf{o}_{0:T-1}) \sum_{\mathbf{o}_T} p(\mathbf{o}_T \mid \mathbf{o}_{0:T-1})\ p(\zeta \mid \mathbf{o}_{0:T})\ p(\zeta \mid \mathbf{o}_{0:T}) \\
&= \sum_{\zeta} \sum_{\mathbf{o}_{0:T-1}} p(\mathbf{o}_{0:T-1}) \sum_{\mathbf{o}_T} \frac{\left(p(\zeta \mid \mathbf{o}_{0:T})\ p(\mathbf{o}_T \mid \mathbf{o}_{0:T-1})\right)^2}{p(\mathbf{o}_T \mid \mathbf{o}_{0:T-1})} \\
&\geq \sum_{\zeta} \sum_{\mathbf{o}_{0:T-1}} p(\mathbf{o}_{0:T-1}) \frac{\left(\sum_{\mathbf{o}_T} p(\zeta \mid \mathbf{o}_{0:T})\ p(\mathbf{o}_T \mid \mathbf{o}_{0:T-1})\right)^2}{\sum_{\mathbf{o}_T} p(\mathbf{o}_T \mid \mathbf{o}_{0:T-1})} \\
&= \sum_{\zeta} \sum_{\mathbf{o}_{0:T-1}} p(\mathbf{o}_{0:T-1})\ p(\zeta \mid \mathbf{o}_{0:T-1})^2 \\
&= \mathbb{E}\left[p(\zeta \mid \mathbf{o}_{0:T-1})\right] = J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0),
\end{aligned}$$

where the inequality follows from application of Sedrakyan's inequality on the innermost sum for each ζ and $\mathbf{o}_{0:T-1}$. When $\mathbf{o}_T$ is continuous, the proof is identical, using the integral version of Sedrakyan's inequality. □

Theorem 3. *An information objective is monotonically nondecreasing.*

Proof. The result follows from the chain rule for mutual information [32], which states that

$$I(\zeta; \mathbf{o}_{1:T} \mid \mathbf{o}_0) = I(\zeta; \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) + \mathbb{E}_{\mathbf{o}_{1:T-1} \mid \mathbf{o}_0}\left[I(\zeta; \mathbf{o}_T \mid \mathbf{o}_{0:T-1}) \mid \mathbf{o}_0\right].$$

This is combined with the non-negativity of mutual information, which states that

$I(\zeta; \mathbf{o}_T \mid \mathbf{o}_{0:T-1}) \geq 0$. Then

$$\begin{aligned} J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) &= I(\zeta; \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) + \mathbb{E}_{\mathbf{o}_{1:T-1} \mid \mathbf{o}_0}\left[I(\zeta; \mathbf{o}_T \mid \mathbf{o}_{0:T-1}) \mid \mathbf{o}_0\right] \\ &\geq I(\zeta; \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \\ &= J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0). \end{aligned}$$

□

Next, we show that the posterior specializations we have outlined preserve monotonicity. Theorem 4 considers a constant domain posterior specialization with value objectives, and Theorem 5 considers variable domain posterior specialization for value and probability objectives.

Theorem 4. *A value objective with a constant domain posterior specialization $\mathcal{V}_{const}$ is monotonically nondecreasing.*

Proof. Consider the potential addition of observation $\mathbf{o}_T$ to $\mathbf{o}_{0:T-1}$. Monotonicity only applies for observations that do not change the value of ζ, so the prior $p(\zeta \mid \mathbf{o}_0)$ does not change. Therefore, for a value query, the following objectives are equal

$$\begin{aligned} &\mathbb{E}_{\mathbf{o}_{1:T-1} \mid \mathbf{o}_0}\left[P\left[\zeta \in \mathcal{V}_{const} \,|\, \mathbf{o}_{0:T-1}\right] \mid \mathbf{o}_0\right] \\ &= \mathbb{E}_{\mathbf{o}_{1:T} \mid \mathbf{o}_0}\left[P\left[\zeta \in \mathcal{V}_{const} \,|\, \mathbf{o}_{0:T}\right] \mid \mathbf{o}_0\right] \\ &= P\left[\zeta \in \mathcal{V}_{const} \mid \mathbf{o}_0\right]. \end{aligned}$$

Therefore, the objective does not change with any additional measurements that do not change ζ, and unintended decreases in reward are not possible. □

To demonstrate monotonicity with variable domain posterior specializations, we will make use of the following lemma.

Lemma 1. *For value and probability objectives, for all $\mathbf{o}_{0:T}$ such that $p(\mathbf{o}_{1:T} \mid \mathbf{o}_0) > 0$,*

$$P\left[\zeta \in \mathcal{V}^*_S(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \leq P\left[\zeta \in \mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right].$$

Proof. For probability objectives, the lemma follows immediately from the definition of $\mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid [0])$ as the set of values of ζ that maximize $p(\zeta \mid \mathbf{o}_{0:T})$ with a fixed cardinality.

Now we consider value objectives. From the identity

$$p(\zeta \mid \mathbf{o}_{0:T-1}) = \mathbb{E}_{\mathbf{o}_T \mid \mathbf{o}_{0:T-1}} \left[p(\zeta \mid \mathbf{o}_{0:T}) \mid \mathbf{o}_{0:T-1} \right],$$

it follows that for all $\mathbf{o}_{0:T}$ that occur with non-zero probability, $p(\zeta \mid \mathbf{o}_{0:T}) > 0$ implies $p(\zeta \mid \mathbf{o}_{0:T-1}) > 0$, and therefore $\mathcal{Z}^+(\mathbf{o}_{0:T}) \subseteq \mathcal{Z}^+(\mathbf{o}_{0:T-1})$.

Now consider any of the largest elements of a fixed cardinality S in $\mathcal{Z}^+(\mathbf{o}_{0:T-1})$. Because $\mathcal{Z}^+(\mathbf{o}_{0:T}) \subseteq \mathcal{Z}^+(\mathbf{o}_{0:T-1})$, each of those elements will either also be the among the largest elements of cardinality S in $\mathcal{Z}^+(\mathbf{o}_{0:T})$, or will not be an element of $\mathcal{Z}^+(\mathbf{o}_{0:T})$. Therefore, for $f(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = \zeta$, all elements of $\mathcal{V}^*_S(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0)$ that are not in $\mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ are not elements of $\mathcal{Z}^+(\mathbf{o}_{0:T})$. We may write this statement as:

$$p(\zeta \mid \mathbf{o}_{0:T}) = 0 \text{ for all } \zeta \in \mathcal{V}^*_S(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \setminus \mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0).$$

Then

$$\begin{aligned}
&P\left[\zeta \in \mathcal{V}^*_S(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \\
&= P\left[\zeta \in \mathcal{V}^*_S(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \cap \mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \\
&\qquad + P\left[\zeta \in \mathcal{V}^*_S(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \setminus \mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \\
&= P\left[\zeta \in \mathcal{V}^*_S(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \cap \mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \\
&\leq P\left[\zeta \in \mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right].
\end{aligned}$$

□

Theorem 5. *A value or probability objective with a variable domain posterior specialization $\mathcal{V}^*_S(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ is monotonically nondecreasing.*

Proof. Consider the potential addition of observation $\mathbf{o}_T$ to $\mathbf{o}_{0:T-1}$. For constant

$\mathbf{o}_{0:T-1}$, by the definition of expectation and marginal probability, we have

$$\mathbb{E}_{\mathbf{o}_T|\mathbf{o}_{0:T-1}}\left[P\left[\zeta \in \mathcal{V}_S^*(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_{0:T-1}\right] \\ = P\left[\zeta \in \mathcal{V}_S^*(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T-1}\right].$$

By lemma 1 we have

$$\begin{aligned} &P\left[\zeta \in \mathcal{V}_S^*(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T-1}\right] \\ &= \mathbb{E}_{\mathbf{o}_T|\mathbf{o}_{0:T-1}}\left[P\left[\zeta \in \mathcal{V}_S^*(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_{0:T-1}\right] \\ &\leq \mathbb{E}_{\mathbf{o}_T|\mathbf{o}_{0:T-1}}\left[P\left[\zeta \in \mathcal{V}_S^*(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_{0:T-1}\right]. \end{aligned}$$

This implies that

$$\begin{aligned} J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) &= \mathbb{E}_{\mathbf{o}_{1:T-1}|\mathbf{o}_0}\left[P\left[\zeta \in \mathcal{V}_S^*(f, \mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T-1}\right] \mid \mathbf{o}_0\right] \\ &\leq \mathbb{E}_{\mathbf{o}_{1:T-1}|\mathbf{o}_0}\left[\mathbb{E}_{\mathbf{o}_T|\mathbf{o}_{0:T-1}}\left[P\left[\zeta \in \mathcal{V}_S^*(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_{0:T-1}\right] \mid \mathbf{o}_0\right] \\ &= \mathbb{E}_{\mathbf{o}_T|\mathbf{o}_0}\left[P\left[\zeta \in \mathcal{V}_S^*(f, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_0\right] \\ &= J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0). \end{aligned}$$

Therefore, an additional observation $\mathbf{o}_T$ that does not affect the true value of ζ can only increase the objective value with posterior specialization. □

Our final theorem, Theorem 6, shows that the addition of a prior specialization will preserve monotonicity.

Theorem 6. *For any objective $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$ such that $\mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right]$ is monotonically nondecreasing, the addition of a prior specialization will preserve monotonicity.*

Proof. We have

$$\begin{aligned} &\mathbb{E}_{\mathbf{o}_{1:T-1}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T-1} \mid \mathbf{o}_0) \mid \mathbf{o}_0, \mathbf{o}_{1:T-1} \in \mathcal{U}(\mathbf{o}_{1:T-1})\right] \\ &\qquad \leq \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0, \mathbf{o}_{1:T-1} \in \mathcal{U}(\mathbf{o}_{1:T-1})\right] \\ &\qquad \leq \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0, \mathbf{o}_{1:T} \in \mathcal{U}(\mathbf{o}_{1:T})\right]. \end{aligned}$$

The first inequality above follows from maximizing $\mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right]$ being monotonically nondecreasing for any distribution over $\mathbf{o}_{1:T}$, including $p(\mathbf{o}_{1:T} \mid \mathbf{o}_0, \mathbf{o}_{1:T-1} \in \mathcal{U}(\mathbf{o}_{1:T-1}))$. The second inequality follows from the definition of $\mathcal{U}(\mathbf{o}_{1:T})$ as the expectation maximizing set with a fixed probability. □

Finally, we draw attention to the fact that even if the query $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ is monotonically nondecreasing, the expression $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$ will *not* be monotonically nondecreasing in general. For example, for certain values of $\mathbf{o}_T$ we may have

$$P\left[\zeta \in \mathcal{V}_{const} \mid \mathbf{o}_{0:T}\right] < P\left[\zeta \in \mathcal{V}_{const} \mid \mathbf{o}_{0:T-1}\right].$$

This means that when a sufficient condition is supplied, certain observations $\mathbf{o}_T$ may reduce $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$ and become further from satisfying $\Delta_{\mathcal{Q}}$. However, since the expectation of $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$ *is* monotonically nondecreasing, $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$ will increase for at least one outcome unless the objective is exactly 0. This makes taking additional observations progress on satisfying the sufficient condition, so that observations without cost will not be ignored.

2.9 Summary

In this chapter, we introduced adaptive sampling that may be used to respond to a class of user-specified queries. We formally introduced a query as a function acting on an instance of the environment model and the locations visited by an agent, an objective that may be modified through the use query specializations, and an optional query threshold that specified a sufficient condition should be met in as few observations as possible. We further showed that optimizing a query objective will lead to an agent that ignores observations unless the query objective is monotonically non-decreasing, and we proved that all queries expressible in our query language are monotonically non-decreasing.

Chapter 3

Planning for Adaptive Sampling with Queries

Given the definition of queries and query-driven adaptive sampling in the previous chapter, we now present a planner that makes decisions on where to gather observations in order to solve query-driven adaptive sampling problems. In this chapter we introduce a Monte Carlo tree search based planner that solves for adaptive sampling strategies that answer all queries expressible in our language. One technical challenge in query-driven adaptive sampling is that we lack closed form expressions for probabilities and information that we will need to evaluate how well a sequence of observations answers a query. In our planner, we propose to compute reward using sample-based estimators of expectation, probability, and mutual information.

Since sample-based estimators can require a large number of samples to converge, one of our innovations is to refine those estimators over the course of rollouts performed during Monte Carlo tree search. Initially, few samples of observations and the variable of interest are taken for a given sequence of actions, to compute reward from a loose estimate of the objective. If a sequence of actions is likely to be optimal, further samples will be taken, to refine the objective estimate for that path. As the algorithm proceeds, most computational effort is spent to determine the optimal objective path from a few candidates.

Applying MCTS to the full set of queries we consider also requires a number of

innovations to be added to MCTS, and this chapter also introduces these changes. We show how to test whether a query has been satisfied by a number of observations, and terminate a rollout early when sufficient information has been gathered. Certain queries also consider only a subset of outcomes, and we show how to estimate reward in these cases using a top percentile of samples gathered.

At the end of this chapter, we test the application of query-driven adaptive sampling on scenarios inspired by the search for hydrocarbon seeps, and emergency responders during a wildfire. We show that query-driven adaptive sampling can be used even for queries that are complex functions of environment variables, and that it outperforms strategies that maximize information about all environment variables.

3.1 Overview of Query-Driven Planning

We now describe an overview of our approach to query-driven adaptive sampling. In existing adaptive sampling problems, the primary source of difficulty is the size of the policy space that must be considered. However, in these existing approaches, reward can typically be computed easily for any sequence of actions, such as through a closed form expression for information. In our case, we face difficulty resulting from the complexity of computing the objective, in addition to the size of the search space.

Complexity in computing reward has two causes. First, the generality permitted by the query function means our algorithm does not have access to a description up front of the prior and posterior distributions $p(\zeta \mid \mathbf{o}_0)$ and $p(\zeta \mid \mathbf{o}_{0:T})$, and cannot rely on assumptions like Gaussianity that are used in other approaches. Dependent on the form of the query, it is not guaranteed that analytic descriptions of distributions of ζ even exist, and so query objectives including expectation of value and mutual information typically cannot be computed in closed form. Further, as functions over random variables, they typically need to be estimated through sampling, an expensive process.

Second, even when the closed form descriptions of $p(\zeta \mid \mathbf{o}_{0:T})$ do exist, such as when it is a discrete distribution, calculating that distribution from the environment

model can be a computationally intensive procedure. To provide an exact result, $p(\zeta \mid \mathbf{o}_{0:T})$ must be computed for each possible value of $\mathbf{o}_{1:T}$ for a given sequence of observation locations $\boldsymbol{x}_{1:T}$, and then for each $\boldsymbol{x}_{1:T}$ considered in the search over policies. When the environment is modeled by a complex machine learning method, computation of $p(\zeta \mid \mathbf{o}_{0:T})$ requires retraining, and this makes it impractical to exactly compute the query objective for every sequence of actions. Furthermore, it is not clear how to define bounds on the objective that would hold for every possible query.

Instead, our approach relies on the fact that, after $\mathbf{o}_{1:t}$ have been observed, $p(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ can be sampled efficiently, and that those samples can be used to estimate $p(\zeta \mid \mathbf{o}_{0:T})$ or quantities like information directly. When imprecise estimates of probability are required, this procedure only requires training $p(\zeta \mid \mathbf{o}_{0:t})$ once for the observations that are taken, and is more efficient than training the posterior $p(\zeta \mid \mathbf{o}_{0:T})$ for every possible $\mathbf{o}_{0:T}$.

The samples can be produced by sampling from the environment model $p(\mathscr{M} \mid \mathbf{o}_{0:t})$, then computing the variable of interest $\zeta = f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \mathscr{M})$. Finally, we sample the observations, based on the attributes $\mathbf{y}(\boldsymbol{x}_t)$ within the sample of the environment model $\mathscr{M}$, according to the distribution $p(\mathbf{o}_t \mid \mathbf{y}(\boldsymbol{x}_t))$. We define and make use of finite sample estimators that can approximate the objective from samples of ζ and $\mathbf{o}_{1:T}$. We apply Monte Carlo tree search (MCTS) in order to search over the space of actions, and we generate samples $\{(\zeta^{(i)}, \mathbf{o}^{(i)}_{t+1:T})\}$ each time a sequence of actions is considered in the search tree. The sample-based estimators are then used to compute a reward based on the query objective $J_{\mathcal{Q}}$.

Rather than expending computational resources on estimating the reward accurately for every sequence of actions considered, we instead generate a coarse estimate for the reward based on relatively few samples. MCTS then guides search to improve the accuracy of estimated reward for sequences of actions based on the likelihood that the coarse estimates are optimal. When a sequence of actions that has been previously reached is reached again, additional samples are taken, and the reward estimate is refined. In this way, reward estimation is made more accurate for those sequences of actions which are believed to be optimal, while not wasting computational resources

on those with poor initial estimates. This is the same basic idea as bandit-based Monte Carlo planning, but our contribution is to embed more complex estimators in the search tree, and to augment MCTS to allow early terminating rollouts and reward derived from a top percentile of samples.

Even when using finite sample estimators for the query objective, it will still be too slow a process to explicitly construct a conditional plan, consisting of a decision tree that branches on possible observations. Instead, we perform Monte Carlo planning online over unconditional search trees. Starting from $\boldsymbol{x}_0$ and observations $\mathbf{o}_0$, we solve for a sequence of actions $a_{0:T-1}$ over the full horizon of the plan that maximizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$. We then execute action a_0, observe $\mathbf{o}_1$, and then solve for an unconditional plan $a_{1:T-1}$ that maximizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{2:T} \mid \mathbf{o}_{0:1})$, repeating this process until T actions have been performed.

The process of executing an adaptive mission using online planning is fairly standard and well understood. Our novel contribution to this process is to detail how query thresholds and prior specializations must be modified in plans generated online in order to ensure the query objective is optimized for the executed plan.

This chapter has the following structure. Section 3.2 describes our MCTS planning approach, and highlights differences from existing applications of MCTS. Section 3.3 describes online planning, how queries must be modified in response to new observations, and an overview of query-driven adaptive sampling. Section 3.4 describes in detail how the finite sample estimators embedded in the search tree are constructed. Finally, Section 3.5 provides experiments comparing query-driven adaptive sampling against approaches maximizing information about all environment variables in the environment, in scenarios motivated by seep search and responding to wildfires.

3.2 Monte Carlo Tree Search with Sample Based Estimators

Monte Carlo tree search (MCTS) with bandit-based action selection [72] was developed to find policies in conditional trees or plans in unconditional trees. In this work, we take the latter approach, and solve for unconditional plans, which are updated online with reward estimated through finite samples.

The basic idea of MCTS is to sample a search tree using a series of *rollouts* from the root state to a leaf state, selecting actions at random. The rollouts are used to generate imprecise estimates of the reward that could be achieved by taking each action in the search tree. These estimates inform which actions should be sampled further, and which action should be selected after sampling is complete.

We apply this approach at each time step t in order to determine a sequence of actions $a_{t:T-1}$. For each sequence of actions, we produce an estimate $\hat{J}$ for the objective $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ from finite samples $\{(\zeta^{(i)}, \mathbf{o}^{(i)}_{t+1:T})\}$ drawn from $p(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$. Paths through the search tree are assigned reward computed from $\hat{J}$, in order to prioritize which paths should be re-sampled. In this section, we discuss how planning is performed using MCTS when given an appropriate objective estimator $\hat{J}$. The exact form of $\hat{J}$ is discussed in Section 3.4.

Our approach differs from standard MCTS in the three following ways:

General estimators in search. The reward gained from taking a sequence of actions is not simply the average of the samples taken. $\hat{J}$ is derived using more complex estimators, including k-NN estimators of mutual information. In addition, when a sufficient condition is specified, the estimators used in the search tree may be used to inform early termination of rollouts, as it will no longer be necessary to rollout to a leaf state.

Refinement of reward for specific action sequences. In MCTS, the reward for visiting a sequence of states in the search tree (corresponding to possible action

decisions) is typically assumed to be known, and finite sample uncertainty arises from estimating the average reward of different sequences of search states. For example, in a Gaussian process model environment with constant kernel hyperparameters, and a Gaussian noise observation model, mutual information $I(\zeta; \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ can be computed in closed form; as can the expectation $\mathbb{E}[\zeta \mid \mathbf{o}_0]$. Sampling with MCTS is then only needed in order to search the space of possible actions. In contrast, in our case the exact reward associated with a sequence of search states is also estimated from a set of samples, so states must be considered multiple times to produce an accurate estimate of the reward.

Expectations over a top percentile of outcomes. Our definition of prior specialization requires maximization of expectation over the top δ_{prior} of outcomes with greatest reward. In order to ensure the correct fraction of outcomes are considered on each branch of the policy, we will rescale δ_{prior} for each online plan.

The following sections review standard MCTS, then discuss the first two additions. We then introduce our online planning approach as a series of MCTS problems that are solved online, and describe how our third addition influences this approach.

3.2.1 Monte Carlo Tree Search

We perform unconditional planning using Monte Carlo tree search over an 'or tree' of search states. For the purpose of discussion, we will assume we are planning from time step 0, but the same principles apply when planning from any time step t. The root state s_0 describes the original state of the vehicle, $\boldsymbol{x}_0$, in addition to all measurements $\mathbf{o}_0$ seen prior to the start of the mission. Each state s_t at time step t describes a specific choice of actions taken by the observing agent. Taking action $a_t \in \mathcal{A}$ leads to state s_{t+1}, which includes $\boldsymbol{x}_{t+1} = d(\boldsymbol{x}_t, a)$ as the new observation location, and a reward $R(s_t, a_t, s_{t+1})$ is received.

The general principle of MCTS-based planning in a tree of search states is shown in Figures 3-1a through 3-1d. Search is performed with a series of *rollouts* from the root state s_0. In each rollout, a series of states from the root state s_0 to a terminal state s_T

are considered, reward is evaluated, and reward information is backpropagated back up the tree to update estimations of future reward.

Each rollout begins with the root state s_0. An action is selected from the state currently under consideration following an action selection rule, and the state resulting from that action is then considered. In our application, the rollout continues until a state on the planning horizon s_T is reached. Reward $r = R(s_t, a_t, s_{t+1})$ from each of the state transitions is observed, and counts of the number of rollouts performed from state s_t with action a_t are recorded as N_{s_t,a_t}, and the total number of rollouts from state s_t is recorded as N_{s_t}. Observed r and N_{s_t,a_t} are used to compute empirical estimates of cumulative future reward $\hat{Q}(s_t, a_t)$ for each state and action pair.

Estimates of future reward are used in the action selection rule for the rollout, guiding search towards actions that are likely to result in high reward. Here, we use the UCB selection rule [7] as proposed in UCT [72], selecting the action from node s_t that satisfies

$$\arg\max_{a_t} = \hat{Q}(s_t, a_t) + \sqrt{\frac{2 \log N_{s_t}}{N_{s_t,a_t}}}. \tag{3.1}$$

This choice of rule balances exploration of the search tree and exploitation of actions that have been explored in order to maximize the cumulative reward observed on all rollouts.

MCTS has been widely used for large discrete search space problems, including adaptive sampling, where the size of the decision space makes searching for a globally optimal solution impractical. Additionally, adaptive sampling problems possess many local optima of poor quality, and it is not easy to escape those suboptimal solutions using optimization based methods.

MCTS alleviates both of these problems by allowing a policy to be generated in anytime manner with continual improvement of the quality of the solution as solution runtime increases.

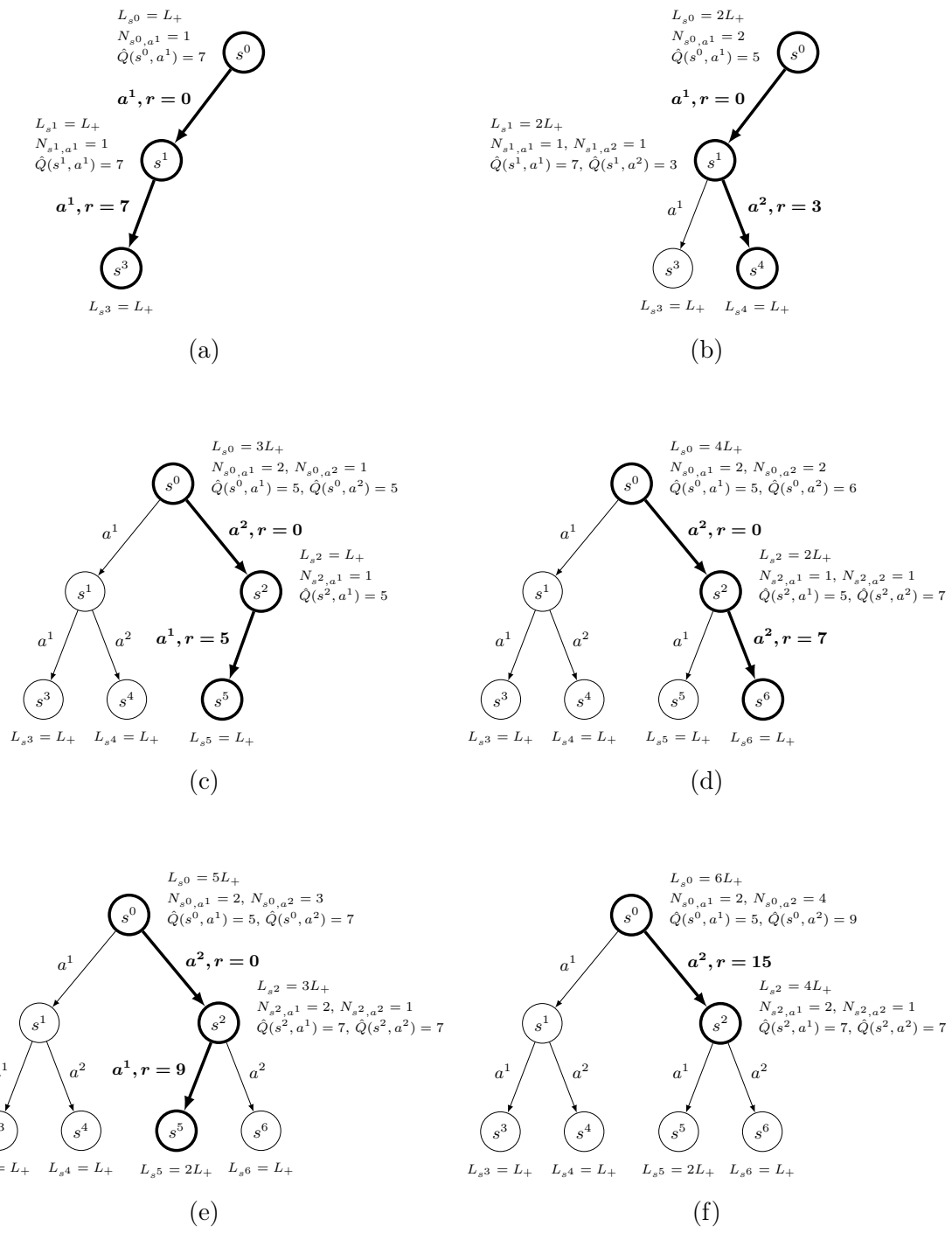


Figure 3-1: Progression of Monte Carlo tree search estimator reevaluation and terminated rollouts in an unconditional tree. States and actions on rollouts are bolded.

3.2.2 MCTS with Estimator Refinement and Terminated Rollouts

Two of the innovations we introduce to standard Monte Carlo tree search above are allowing complex objective estimators to be refined with additional samples over the course of search, and allowing rollouts to be terminated before reaching the planning horizon when sufficient reward is achieved.

An algorithmic description of our MCTS approach is given in Algorithm 1. Rollouts are performed with successive calls to `SampleState`, beginning at the root node. When a state s_t is sampled, if a sufficient condition has been specified for a query or s_t is a leaf state, an estimation $\hat{J}(s_t, \mathcal{Q})$ of the objective value for observations $\mathbf{o}_{1:t}$ is computed through the use of a routine `RefineEstimator` on line 6. An immediate reward is then computed from the estimate using the procedure `ImmediateReward` on line 7. When a sufficient condition is specified, the rollout will end immediately if $\hat{J}(s_t, \mathcal{Q})$ exceeds the query threshold $\Delta_{\mathcal{Q}}$. Checking this condition is performed within the routine `ContinueRollout` on line 8. When the rollout is allowed to continue, a child state s_{t+1} is selected according to UCT's selection rule on line 9, then s_{t+1} is sampled. After s_{t+1} has been sampled, the future reward estimate $\hat{Q}(s_t, a_t)$ is updated, and the reward achieved from s_t onwards is returned from `SampleState`.

We now discuss how each of these procedures operates in order to lead to desired behavior.

Estimator Refinement

`RefineEstimator` takes additional samples of the environment model $\mathscr{M}$, which are then used to produce samples of the variable of interest and observations. These are then used to compute an updated value of the estimators described in Section 3.4.

This procedure is necessary because the average reward gained from visiting a state is not simply the average of the samples taken at that state when using complex objective estimators. Each time a state s_t (resulting from actions $a_{0:t-1}$) is visited, L_+ additional samples of ζ and $\mathbf{o}_{1:t}$ are added for use in computation and refinement

Algorithm 1: MCTS

Input : Initial state s_0, planning horizon T, max planning time τ, query $\mathcal{Q}$
Output: Next action to execute

loop until planning time τ reached
  `SampleState(`s_0, $\mathcal{Q}$`)`
return $\arg\max_a \hat{Q}(s_0, a)$

Procedure `SampleState(`s_t, $\mathcal{Q}$`)`
  if sufficient condition $\Delta_{\mathcal{Q}}$ in query or $t = T$ **then**
    `RefineEstimator(`s_t, L_+, $\mathcal{Q}$`)`
  $r \leftarrow$ `ImmediateReward(`s_t, $\mathcal{Q}$`)`
  if $t \neq T$ and `ContinueRollout(`r, $\mathcal{Q}$`)` **then**
    $a \leftarrow \arg\max_{a_t} \hat{Q}(s_t, a_t) + \sqrt{\frac{2 \log N_{s_t}}{N_{s_t, a_t}}}$
    $s_{t+1} \leftarrow$ child state with $\boldsymbol{x}_{t+1} = d(\boldsymbol{x}_t, a)$
    $q \leftarrow$ `SampleState(`s_{t+1}, $\mathcal{Q}$`)`
    $\hat{Q}(s_t, a) \leftarrow \frac{N_{s_t,a}\ \hat{Q}(s_t,a)+q}{N_{s_t,a}+1}$
    $N_{s_t,a} \leftarrow N_{s_t,a} + 1$
    $r \leftarrow r + q$
  $N_{s_t} \leftarrow N_{s_t} + 1$
  return r

of the estimator. This procedure is shown through Figure 3-1. After visiting s_t a total of N_{s_t} times, the total number of samples taken is $L_{s_t} = N_{s_t} L_+$, all of which are used to produce an updated estimate $\hat{J}(s_t, \mathcal{Q})$.

In close analogy to existing MCTS approaches, we could choose $L_+ = 1$, taking a single sample each time a state is visited. However, evaluating $\hat{J}(s_t, \mathcal{Q})$ is not necessarily a simple operation, and is not evaluated on a single sample alone, but a collection of samples. In the case of k-NN estimators, k nearest neighbors of every sample must be found, typically through the use of a structure such as a KD-tree. Incrementally adding new samples to a KD-tree causes it to quickly become unbalanced, and it must be reconstructed. From this perspective, it is advantageous to visit a state fewer times with a higher value of L_+, in order to avoid repeatedly constructing data structures needed to compute $\hat{J}(s_t, \mathcal{Q})$.

On the other hand, we could use a very large number of samples on each rollout, and gain an accurate evaluation of the reward for every sequence of actions. But this method is also unnecessarily wasteful when we are using computational resources to

refine estimates for suboptimal decisions. Our approach is to make use of a noisy initial estimate with a moderate number of samples, typically with L_+ on the scale of a few hundred samples. We then use reward computed from that estimate to guide whether taking further samples is justified. Frequently, the initial estimate is sufficient to tell that a sequence of actions is unlikely to be optimal, even if the true reward is not known exactly.

It is possible to reduce the number of samples needed by reusing the samples that have been taken in multiple branches. The samples $(\zeta^{(i)}, \mathbf{o}_{1:t}^{(i)})$ after visiting $\boldsymbol{x}_{1:t}$ are generated from a sample $\mathscr{M}^{(i)}$. The sample $\mathscr{M}^{(i)}$ does not depend on the locations explored, so when generating $(\zeta^{(j)}, \mathbf{o}_{1:t}^{(j)})$ after visiting some $\boldsymbol{x}'_{1:t} \neq \boldsymbol{x}_{1:t}$, we may re-use the sample $\mathscr{M}^{(i)}$. As a result, we cache all model samples $\mathscr{M}^{(i)}$ generated over the course of search.

An overview of this procedure is provided in Algorithm 2. The number of samples drawn from the model is stored as $L_{\mathscr{M}}$, and the number of samples taken for state s_t is stored as L_{s_t}. On line 2, it is checked whether `RefineEstimator` is called for a state s_t with $L_{\mathscr{M}} < L_{s_t} + L_+$. If $L_{\mathscr{M}}$ is not large enough, then an additional L_{update} samples are taken in lines 3 through 7. Otherwise, the existing samples $\mathscr{M}^{(L_{s_t}:L_{s_t}+L_+)}$ are used to generate the L_+ additional query and observation samples. It is not necessary that $L_{update} = L_+$; depending on the time taken to load the environment model into memory and any computations required to prepare sampling, it may be more beneficial to gather environment model samples in larger batches, using $L_{update} > L_{sample}$.

The remainder of `RefineEstimator` shows the additional L_+ samples of observations and query variables of interest being generated in the for loop on line 8, before and updated objective estimate is generated using those samples.

Immediate Reward and Terminated Rollouts

The procedure `ImmediateReward` then provides a means to compute the reward $R(s_{t-1}, a_{t-1}, s_t)$ from $\hat{J}(s_t, \mathcal{Q})$. The reward may be considered an estimate for the top level objective W. A decision is then made about whether to continue the rollout

Algorithm 2: Reward Computation

Procedure `RefineEstimator`$(s_t, L_+, \mathcal{Q})$
  if $L_{\mathscr{M}} < L_{s_t} + L_+$ **then**
    for $i = L_{\mathscr{M}} + 1, \ldots, L_{\mathscr{M}} + L_{update}$
      $\mathbf{c}^{(i)}, \mathcal{E}^{(i)}, \boldsymbol{\lambda}^{(i)} \leftarrow$ `SampleDist`$(p(\mathbf{c}, \mathcal{E}, \boldsymbol{\lambda}))$
      $\mathbf{y}^{(i)} \leftarrow$ `SampleDist`$(p(\mathbf{y} \mid \mathbf{c}^{(i)}, \mathcal{E}^{(i)}, \boldsymbol{\lambda}^{(i)}))$
      $\mathscr{M}^{(i)} \leftarrow \langle \mathbf{y}^{(i)}, \mathbf{c}^{(i)}, \mathcal{E}^{(i)}, \boldsymbol{\lambda}^{(i)} \rangle$
    $L_{\mathscr{M}} \leftarrow L_{\mathscr{M}} + L_{update}$
  for $j = L_{s_t} + 1, \ldots, L_{s_t} + L_+$
    $\mathbf{o}_{1:t}^{(j)} \leftarrow$ `SampleDist`$(p(\mathbf{o}_{1:t} \mid \mathbf{y}^{(j)}(\boldsymbol{x}_{1:t})))$
    $\zeta^{(j)} \leftarrow f_{\mathcal{Q}}(\boldsymbol{x}_{1:t}, \mathscr{M}^{(j)})$
  $L_{s_t} \leftarrow L_{s_t} + L_+$
  $\hat{J}(s_t, \mathcal{Q}) \leftarrow$ `ObjectiveEstimator`$(J_{\mathcal{Q}}, \zeta^{(1:L_{s_t})}, \mathbf{o}_{1:t}^{(1:L_{s_t})}, L_{s_t})$

Procedure `ImmediateReward`$(s_t,\ \mathcal{Q})$
  if sufficient condition $\Delta_{\mathcal{Q}}$ in $\mathcal{Q}$ **then**
    if $\hat{J}(s_t, \mathcal{Q}) \geq \Delta_{\mathcal{Q}}$ **then**
      return $-\left(t - 1 + \frac{\Delta_{\mathcal{Q}} - \hat{J}(s_t, \mathcal{Q})}{\hat{J}(s_t, \mathcal{Q}) - \hat{J}(s_{t-1}, \mathcal{Q})}\right)$
    else if $t = T$ **then**
      return $-\left(T + 1 - \frac{\hat{J}(s_t, \mathcal{Q})}{\Delta_{\mathcal{Q}}}\right)$
    else
      return 0
  else
    if $t = T$ **then**
      return $\hat{J}(s_t, \mathcal{Q})$
    else
      return 0

Procedure `ContinueSampling`$(r,\ \mathcal{Q})$
  if sufficient condition $\Delta_{\mathcal{Q}}$ in $\mathcal{Q}$ and $\hat{J}(s_t, \mathcal{Q}) \geq \Delta_{\mathcal{Q}}$ **then**
    return False
  else
    return True

or immediately end it in the procedure `ContinueRollout`. Both `ImmediateReward` and `ContinueRollout` are described in Algorithm 2.

`ImmediateReward` and `ContinueRollout` change behavior dependent on whether a sufficient condition is specified in the query. When a sufficient condition is not specified, the objective W is to maximize the query objective over a fixed number of T actions. Since the estimator $\hat{J}(s_t, \mathcal{Q})$ computes an estimate of $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:t} \mid \mathbf{o}_0)$, including all observations up to state s_t, it is only necessary for the estimator to be computed for leaf states in the search tree, which will be used to determine reward for a full sequence of actions. As a result, `ImmediateReward` returns 0 before reaching a leaf state in line 25. At a leaf state s_T, $\hat{J}(s_T, \mathcal{Q})$ is returned on line 23. There is no need to terminate a rollout early without a sufficient conditon, so `ContinueRollout` always returns True on line 30.

On the other hand, when a sufficient condition $\Delta_{\mathcal{Q}}$ is specified, then in MCTS we solve for the shortest sequence of actions for which $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:t} \mid \mathbf{o}_0) \geq \Delta_{\mathcal{Q}}$. In this case, it is necessary to compute an objective estimate at every state, in order to determine whether the threshold $\Delta_{\mathcal{Q}}$ has been reached by $\mathbf{o}_{1:t}$. To match the objective W, the reward to be maximized is taken as the negative of the number of actions taken to reach $\Delta_{\mathcal{Q}}$. However, it is typically favorable to prioritize missions that achieve a higher value for $J_{\mathcal{Q}}$ with the same mission length, and from the perspective of the planner, it is less likely that a state with a larger estimate $\hat{J}(s_t, \mathcal{Q})$ fails to truly satisfy $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:t} \mid \mathbf{o}_0) \geq \Delta_{\mathcal{Q}}$ due finite sample errors in the estimator. To favor states with larger $\hat{J}(s_t, \mathcal{Q})$ among those that achieve the sufficient condition at the same depth, `ImmediateReward` returns an estimate of the non-integer number of time steps needed to reach $\Delta_{\mathcal{Q}}$ on line 16, computed using a linear interpolation between the objectives at s_{t-1} and s_t. When $\hat{J}(s_t, \mathcal{Q})$ is significantly higher than $\Delta_{\mathcal{Q}}$, or $\hat{J}(s_{t-1}, \mathcal{Q})$ is close to $\Delta_{\mathcal{Q}}$, the reward returned is closer to $-(t-1)$ than $-t$, favoring the actions that led to s_t in the search tree.

There is no value in continuing to sample deeper in the search tree once the query threshold is achieved at s_t. Rollouts are terminated once the sufficient condition is reached by returning False from `ContinueRollout` on line 28. When a rollout

is terminated at state s_t, the reward returned consists only of the immediate reward $R(s_{t-1}, a_{t-1}, s_t)$. The count N_{s_t} of rollouts from s_t is *not* incremented, since the rollout does not continue on to actions from s_t. This means that the process of choosing an action from s_t, which may still occur on later samples dependent on future estimates of $\hat{J}(s_t, \mathcal{Q})$, is not affected by terminating a rollout early. This procedure is shown in Figure 3-1f, where after a sufficiently high value of $R(s^0, a^2, s^2)$ is received, the rollout is not continued past s^2. $\hat{Q}(s^2, a^1)$ and $\hat{Q}(s^2, a^2)$ are not changed, but $\hat{Q}(s^0, a^2)$ is.

There remains the question of what to return from states where $\hat{J}(s_t, \mathcal{Q}) < \Delta_{\mathcal{Q}}$. For $t \neq T$, it is possible that a successor state does achieve the sufficient condition, so `ImmediateReward` returns 0 on line 20 and `ContinueRollout` returns True on line 30. Using a reward of 0 ensures that $\hat{Q}$ includes only the reward obtained from the successor state where the sufficient condition is achieved.

When $\Delta_{\mathcal{Q}}$ is not reached by a leaf state s_T, we do not want to eliminate s_T from the search tree, because it is possible that a more accurate estimate using additional samples would reveal that $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) \geq \Delta_{\mathcal{Q}}$ is satisfied. But the reward should be lower than that of any state that does achieve the sufficient condition. A logical approach would be to return the negative of an extrapolation of the number of actions required to reach an objective of $\Delta_{\mathcal{Q}}$, computed from $\hat{J}(s_T, \mathcal{Q})$. However, this number can be a negative number with a high magnitude for sequences of actions $a_{0:T-1}$ with $\hat{J}(s_T, \mathcal{Q})$ close to 0. We empirically found that since the reward is propagated all the way to the root state, sampling a single sequence of actions with $\hat{J}(s_T, \mathcal{Q}) \approx 0$ would cause $\hat{Q}(s_0, a_0)$ to become a large negative number, and action a_0 would never be sampled again from the root state by UCT's action selection rule. This is undesirable, because other sequences of actions starting with a_0 that do satisfy the sufficient condition may exist, and perhaps may have already be sampled.

Instead, we return a reward in the range $[-T-1, -T]$, using a formula

$$R(s_{T-1}, a_{T-1}, s_T) = -T - 1 + \frac{\hat{J}(s_T, \mathcal{Q})}{\Delta_{\mathcal{Q}}} \tag{3.2}$$

on line 18. This formula maps states with $\hat{J}(s_T, \mathcal{Q})$ close to $\Delta_{\mathcal{Q}}$ to a reward close to

$-T$, while states with $\hat{J}(s_T, \mathcal{Q}) \approx 0$ return $-T - 1$.

3.3 Online Planning

In order to make our plan adaptive to observations received as the plan executes, we perform online planning, where Monte Carlo Tree Search is run after each observation. We now discuss online planning in detail, including a justification of why we perform planning online, and how objectives need to change after each observation in order to appropriately generate a plan for the query.

When planning online, after having taken observation $\mathbf{o}_{t-1}$, the query is updated to be a modified $\mathcal{Q}_{t-1}$, based on the observations that were received. The query is changed to optimize an objective that is conditioned on observations $\mathbf{o}_{0:t-1}$. As we will discuss in this section, the query is also modified to rescale query thresholds and prior specializations in order to ensure desired behavior. MCTS then generates a plan consisting of actions $a_{t-1}, \ldots, a_{T-1}$. Action a_{t-1} is performed, and the observation $\mathbf{o}_t$ is observed. The environment model is then updated, based on the observations $\mathbf{o}_{0:t}$, and the next plan is generated from state s_t with a query $\mathcal{Q}_t$. Requiring planning to be done online may limit the total time made available for planning, but we note that MCTS allows a plan to be produced in an anytime manner, so that a plan is always available.

Our approach expands upon prior work in online MCTS by ensuring that query thresholds and prior specializes are satisfied within an online planning framework. Generating the correct behavior requires changing the query threshold and prior specialization confidence levels in response to observations as they are received.

3.3.1 Changes in Objectives During Online Planning

When we produce a new plan after receiving an observation, the objective changes. For example, when planning at the start of the mission, where the only observations are $\mathbf{o}_0$, the objective used is $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$, which is conditioned only on $\mathbf{o}_0$ and averages over $\mathbf{o}_{1:T}$ that could be observed. Later, after observing $\mathbf{o}_{1:t}$, we produce a

new plan that optimizes the objective $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$, which is now conditioned on all of the $\mathbf{o}_{0:t}$ that are observed and only averages over $\mathbf{o}_{t+1:T}$. In this section, we clarify the relationship between optimizing $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ in response to every possible $\mathbf{o}_{1:t}$ and the overall objective for the mission $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$.

We will compare fully policies that maximize $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ against policies that maximize $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ for a given value of $\mathbf{o}_{0:t}$. If the behavior of our planning algorithm is correct, the policy that maximizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ will perform the same actions that would be performed by the policy that maximizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ after $\mathbf{o}_{1:t}$ have been observed. Our goal is to ensure that this property holds even with changes in the objective during online planning.

We will show that for value, probability, and information objectives, follow the actions of a policy that maximizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ after every $\mathbf{o}_{0:t}$ for all t is equivalent to following a policy that maximizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ for the duration of the mission. This result provides some justification for why it is appropriate to optimize $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ during online planning. However, since we solve for the unconditional *plan* that maximizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ and not the conditional policy, the result does *not* mean that the policy that is followed by performing online planning optimizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$.

To show that the policies are equivalent, first note that ζ is computed using the full history of actions $\boldsymbol{x}_{1:T}$ for all plans, so the interpretation of ζ is unchanged between observations. Next, consider an agent following the policy that maximizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$. After having received any set of observations $\mathbf{o}_{0:t}$ and entering state s_t, the remainder of the policy from s_t will maximize $\mathbb{E}_{\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}}[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t}]$. If it holds that

$$\max_{\pi} \ \mathbb{E}_{\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}}[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t}] = \max_{\pi} \ J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}), \tag{3.3}$$

for any $p(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$, then those remaining actions result in the same objective value as a policy that maximizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ from s_t. If (3.3) further holds for all $\mathbf{o}_{0:t}$ and all t, then following the first actions of policies that maximize $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$

at each t must result in a policy that maximizes $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$.

For value and probability objectives, we can show that (3.3) is true because

$$\mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t}\right] = J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}). \tag{3.4}$$

For value objectives

$$\begin{aligned}
\mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t}\right] &= \mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[\mathbb{E}_{\zeta|\mathbf{o}_{0:T}}\left[\zeta \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_{0:t}\right] \\
&= \mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right] \\
&= J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})
\end{aligned} \tag{3.5}$$

so the objectives align and (3.3) holds. Likewise, for probability objectives

$$\begin{aligned}
\mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t}\right] &= \mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[\mathbb{E}_{\zeta|\mathbf{o}_{0:T}}\left[p(\zeta \mid \mathbf{o}_{0:T}) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_{0:t}\right] \\
&= \mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right] \\
&= J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}).
\end{aligned} \tag{3.6}$$

In both cases, these results held because $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) = g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$. Furthermore, posterior specializations depend on each full observation sequence $\mathbf{o}_{0:T}$, and so can be applied without modification.

The situation is slightly different for information objectives, because the original information $I(\zeta; \mathbf{o}_{1:T} \mid \mathbf{o}_0)$ depends on the prior probabilities $p(\zeta \mid \mathbf{o}_0)$ and $p(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$. This means that $I(\zeta; \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \neq \mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t}\right]$. Instead, we have

$$\begin{aligned}
&\mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t}\right] \\
&= \mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[\mathbb{E}_{\zeta|\mathbf{o}_{0:T}}\left[lr(\zeta; \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_{0:t}\right] \\
&= \mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[\mathbb{E}_{\zeta|\mathbf{o}_{0:T}}\left[lr(\zeta; \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_{0:t}\right] + \mathbb{E}_{\zeta|\mathbf{o}_{0:t}}\left[lr(\zeta; \mathbf{o}_{1:t} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t}\right] \\
&= I(\zeta; \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) + g(\mathbf{o}_{1:t} \mid \mathbf{o}_0) \\
&= J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) + g(\mathbf{o}_{1:t} \mid \mathbf{o}_0).
\end{aligned} \tag{3.7}$$

Since maximization of $\mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t}]$ is equal to maximization of the sum of $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$ and a constant term depending only on prior observations $\mathbf{o}_{0:t}$, then (3.3) continues to hold.

When a sufficient condition is not specified, the constant term can be safely ignored when replanning with observations $\mathbf{o}_{0:t}$. However, it must be explicitly handled in the objective when comparing against $\Delta_{\mathcal{Q}}$. Information objectives do not accept posterior specializations, so there is no need for them to handle the constant offset.

3.3.2 Treatment of Sufficient Conditions in Online Planning

There are two types of decisions that need to be made when planning with a sufficient condition; deciding when to terminate rollouts during planning because the query threshold is reached by the plan, and making the decision to end the mission after observations have been received. We described early terminating rollouts in our discussion of MCTS, but we now discuss how these behaviors change during online planning.

First reconsider early terminating rollouts. When planning from an initial state, $\hat{J}(s_t, \mathcal{Q})$ is an estimate for $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:t} \mid \mathbf{o}_0)$, and we described that rollouts should be terminated at any state s_t such that $\hat{J}(s_t, \mathcal{Q}) \geq \Delta_{\mathcal{Q}}$. Now consider planning from state $s_{t'}$, after having observed $\mathbf{o}_{0:t'}$. In this search tree, $\hat{J}(s_t, \mathcal{Q})$ is now an estimate for the objective $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t'+1:t} \mid \mathbf{o}_{0:t'})$. For value and probability objectives, the results from Section 3.3.1 showed that $\mathbb{E}_{\mathbf{o}_{t'+1:t}|\mathbf{o}_{0:t'}}[g(\mathbf{o}_{1:t} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t'}] = J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t'+1:t} \mid \mathbf{o}_{0:t'})$. This means that the new objective is the expectation of $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$ conditioned on the observations that have been gathered, and it is suitable to compare the objective against $\Delta_{\mathcal{Q}}$, which was a limit on $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$. As a result, for value and probability objectives, the condition that rollouts terminate when $\hat{J}(s_t, \mathcal{Q}) \geq \Delta_{\mathcal{Q}}$ holds for all plans generated online.

Next, to determine whether a mission should end after planning from state s_{t-1}, executing action a_{t-1}, and observing $\mathbf{o}_t$, we must determine whether to end the mission. A query threshold $\Delta_{\mathcal{Q}}$ dictates that the mission should end if $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0) \geq \Delta_{\mathcal{Q}}$. Therefore, we require an additional estimator for g, which we refer to as $\hat{g}(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$.

If $\hat{g}(\mathbf{o}_{1:t} \mid \mathbf{o}_0) \geq \Delta_{\mathcal{Q}}$ is found to hold, then the sufficient condition is deemed to have been satisfied, and the mission immediately ends. For value and probability objectives, the fact that $\hat{J}(s_t, \mathcal{Q})$ is an estimate for the expectation of $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$ will mean that $\hat{g}(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$ can be naturally evaluated using the samples taken when planning from state s_{t-1}, and its form naturally follows from the computation of $\hat{J}(s_t, \mathcal{Q})$. We therefore defer discussion on how to compute $\hat{g}(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$ until Section 3.4, where the estimators for $\hat{J}(s_t, \mathcal{Q})$ are introduced.

When using an information objective, we showed $\mathbb{E}_{\mathbf{o}_{t'+1:t}|\mathbf{o}_{0:t'}}[g(\mathbf{o}_{1:t} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t'}] = J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t'+1:t} \mid \mathbf{o}_{0:t'}) + g(\mathbf{o}_{1:t'} \mid \mathbf{o}_0)$, and this result makes planning online more complex. Since the objective $J_{\mathcal{Q}}(\zeta, \mathbf{o}_t \mid \mathbf{o}_{0:t-1})$ when planning from state $t-1$ is no longer the expectation of $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$, the samples used to estimate $\hat{J}(s_t, \mathcal{Q})$ can only be used to estimate $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0) - g(\mathbf{o}_{1:t-1} \mid \mathbf{o}_0)$ instead of $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$. As a result, after observing $\mathbf{o}_t$, we will need to use additional samples cached from earlier plans in order to generate the estimate $\hat{g}(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$. Details are given in Section 3.4.

The rule for terminating rollouts must also change when using an information objective, due to fact that $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t'+1:t} \mid \mathbf{o}_{0:t'})$ differs from an expectation of $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$ by $g(\mathbf{o}_{1:t'} \mid \mathbf{o}_0)$, and so comparison against $\Delta_{\mathcal{Q}}$ is not suitable. To solve this problem, we will use the fact that when planning from a state $s_{t'}$ with a sufficient condition and an information objective, we will have already computed the constant term $\hat{g}(\mathbf{o}_{1:t'} \mid \mathbf{o}_0)$ to determine whether the mission should have ended after observing $\mathbf{o}_{t'}$. We then determine whether to terminate rollouts at any state s_t using a newly defined threshold $\Delta_{\mathcal{Q},t'}$ for planning from $s_{t'}$. Using the previous estimate, we define

$$\Delta_{\mathcal{Q},t'} = \Delta_{\mathcal{Q}} - \hat{g}(\mathbf{o}_{1:t'} \mid \mathbf{o}_0), \tag{3.8}$$

and terminate rollouts once $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{t'+1:t} \mid \mathbf{o}_{0:t'}) \geq \Delta_{\mathcal{Q},t'}$.

3.3.3 Rescaling of Prior Specializations in Online Planning

Performing planning online will also introduce complexity into how prior specializations are computed. During the course of online planning, we must also rescale a

prior specialization level δ_{prior} to maintain correct behavior.

To see why, consider using online planning to solve for a policy with a prior specialization of δ_{prior}. That is,

$$\max_{\pi, \mathcal{U}} \ \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0} \left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0, \mathbf{o}_{1:T} \in \mathcal{U} \right] \text{ s.t. } P\left[\mathbf{o}_{1:T} \in \mathcal{U} \mid \mathbf{o}_0 \right] \geq \delta_{prior}.$$

For the purpose of discussion, consider the and/or tree of outcomes given in Figure 3-2. There are no prior observations, $T = 2$, $\delta_{prior} = 0.5$, there is one possible action at state s^0 and two actions available at s^1 and s^2, and all actions have two equally likely outcomes. The observations received and the value of $g(\mathbf{o}_{0:2})$ are displayed beneath each leaf state.

Since we plan over an or tree, we do not generate the tree in Figure 3-2 explicitly, instead $\mathbf{o}_1^1$ and $\mathbf{o}_1^2$ are in a single state resulting from a^1, and $\mathbf{o}_2^1$, $\mathbf{o}_2^2$, $\mathbf{o}_2^5$, and $\mathbf{o}_1^6$ are in a single state resulting from the sequence $[a^1, a^1]$. Nonetheless, the plan developed online after receiving the first observation will depend on whether $\mathbf{o}_1^1$ or $\mathbf{o}_1^2$ is received. As a result, execution will follow one of the branches shown in this tree. We wish to analyze the behavior of our algorithm across different outcomes, and so we visualize the full and/or tree here.

To simplify analysis, we first ignore finite sample limits, and assume we have access to exact probabilities and rewards. When planning from s_t, we generate states below s_t, and condition on observations $\mathbf{o}_{1:t}$ that were observed. We search for the sequence of actions that maximizes

$$\mathbb{E}_{\mathbf{o}_{t+1:2}|\mathbf{o}_{1:t}} \left[g(\mathbf{o}_{1:2}) \mid \mathbf{o}_{1:t}, \mathbf{o}_{1:2} \in \mathcal{U}_t \right] \text{ s.t. } P\left[\mathbf{o}_{1:2} \in \mathcal{U}_t \mid \mathbf{o}_{1:t} \right] \geq \delta_{prior,t}$$

for some $\mathcal{U}_t$ and $\delta_{prior,t}$. When planning first starts from s^0, $\mathcal{U}_0 = \mathcal{U}$, and $\delta_{prior,0} = \delta_{prior}$, and we find the optimal sequence of actions to be $[a^1, a^1]$ with conditional expectation 12.

In this example, the optimal policy is to select action a^1 from s^1, and the action selected from s^2 is irrelevant. When performing unconditional planning from state s^0, we would find the optimal action sequence to be $[a^1, a^1]$. The set of observations

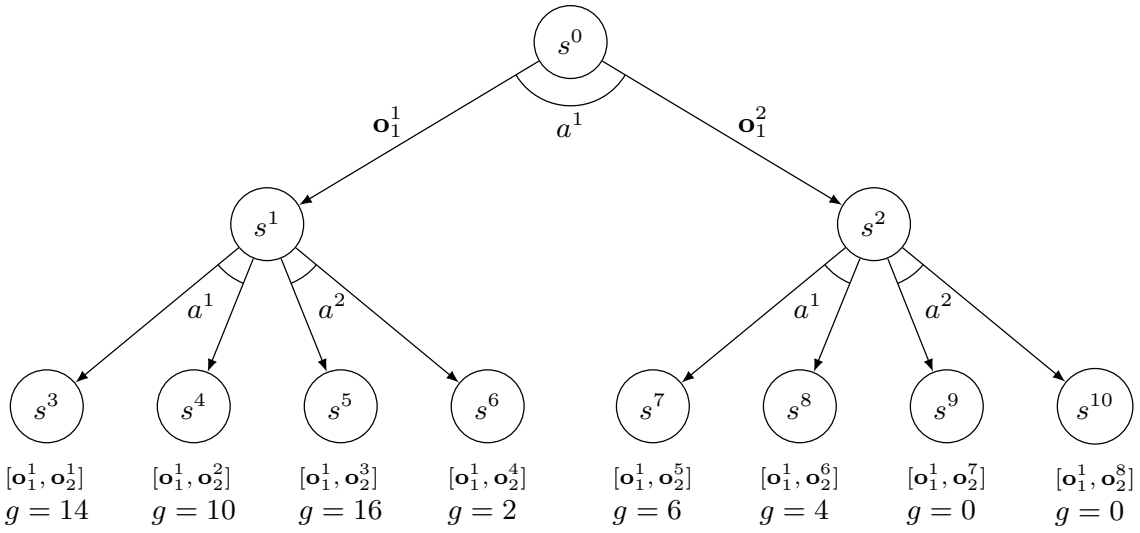


Figure 3-2: Example and/or search tree to demonstrate the effect of online planning under a prior specialization. All actions have two equally likely outcomes, indicated by arcs.

with probability at least 0.5 that results in the highest conditional expectation is $\mathcal{U} = \{[\mathbf{o}_1^1, \mathbf{o}_2^1], [\mathbf{o}_1^1, \mathbf{o}_2^2]\}$, with $\mathbb{E}_{\mathbf{o}_{1:2}}[g(\mathbf{o}_{1:2}) \mid \mathbf{o}_{1:2} \in \mathcal{U}] = 12$. Letting $\mathcal{U}^i$ denote the elements of $\mathcal{U}$ that begin with the observations seen up to s^i, we have $\mathcal{U}^1 = \{[\mathbf{o}_1^1, \mathbf{o}_2^1], [\mathbf{o}_1^1, \mathbf{o}_2^2]\}$ and $\mathcal{U}^2 = \emptyset$.

Now assume that after executing a^1 from s^0, the agent observes $\mathbf{o}_1^1$ and enters s^1. If we were to replan from s^1 with $\delta_{prior,1} = 0.5$, we would now find that a^2 is the best action, because we find that the set of observations with 50% probability conditioned on $\mathbf{o}_1^1$ and highest reward is $\mathcal{U}_1 = \{[\mathbf{o}_1^1, \mathbf{o}_2^3]\}$, with conditional expectation 16. The result when planning from s^2 is also different, because the best action is a^1, when in optimal policy it does not matter. The issue here is that the outcomes define $\mathcal{U}$ in the optimal policy both result from s^1. However, planning from s^1 and s^2 with $\delta_{prior,1} = 0.5$ considers only the best single result from each state, leading to suboptimal actions. The optimal policy can be executed using unconditional planning by planning from s^1 using $\delta_{prior,1} = 1.0$, and from s^2 using $\delta_{prior,1} = 0$, which result in $\mathcal{U}_1 = \{[\mathbf{o}_1^1, \mathbf{o}_2^1], [\mathbf{o}_1^1, \mathbf{o}_2^2]\}$ and $\mathcal{U}_1 = \emptyset$ respectively.

Correct behavior from s_1, that considers the expectation of $g(\mathbf{o}_{1:2})$ over outcomes consistent with the plan found at s_0, assigns the probability $\delta_{prior,1}$ to $\mathcal{U}_1$ based on

how frequently occurrences of $\mathbf{o}_1$ also occur in the δ_{prior}-optimal set of outcomes $\mathcal{U}$ in the top level plan from s_0. When many observations that start with $\mathbf{o}_1$ appear in the top level $\mathcal{U}$, then we should maximize expectation computed over more outcomes that follow $\mathbf{o}_1$, and therefore increase $\delta_{prior,1}$. More precisely, $\delta_{prior,1}$ constrains $P\left[\mathbf{o}_{1:2} \in \mathcal{U}_1 \mid \mathbf{o}_1\right]$. To maintain consistency with the optimal actions found at the root, $P\left[\mathbf{o}_{1:2} \in \mathcal{U}_1 \mid \mathbf{o}_1\right]$ should be set to be the closest match to $P\left[\mathbf{o}_{1:2} \in \mathcal{U}_0 \mid \mathbf{o}_1\right]$. Therefore, we set $\delta_{prior,1} = P\left[\mathbf{o}_{1:2} \in \mathcal{U}_0 \mid \mathbf{o}_1\right]$. In the context of the example above, all observations that contain $\mathbf{o}_1^1$ were elements of $\mathcal{U}$ when planning from s^0, so all observations that follow from $\mathbf{o}_1^1$ should be considered as part of $\mathcal{U}_1$ when planning from s^1, resulting in $\delta_{prior,1} = 1.0$.

To formalize this idea, we plan from state s_t with known observations $\mathbf{o}_{0:t}$ using a prior specialization level $\delta_{prior,t}$. $\delta_{prior,t}$ is the probability of set $\mathcal{U}_t$, which consists of the observations that maximize $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$ and begin with the known $\mathbf{o}_{1:t}$. Starting with $\delta_{prior,0} = \delta_{prior}$ as specified in the original objective, after planning, the action sequence $a_{t:T-1}$ is found to maximize $\mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_{0:t}, \mathbf{o}_{1:T} \in \mathcal{U}_t\right]$. Action a_t is then executed, and observation $\mathbf{o}_{t+1}$ is observed. We then set $\delta_{prior,t+1}$ for the next sequence of actions using

$$\delta_{prior,t+1} = P\left[\mathbf{o}_{1:T} \in \mathcal{U}_t \mid \mathbf{o}_{0:t+1}\right]. \tag{3.9}$$

So far, we have assumed that probabilities of all observations are available to the planner, but in query-driven adaptive sampling we use samples of observations obtained from rollouts. When planning from state s_t, we calculate the reward for a path from s_t to s_T using a set of independent observations $\{\mathbf{o}_{1:T}^{(i)}\}$ that start with known $\mathbf{o}_{1:t}$ and with the remainder drawn from the distribution $p(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$. We then construct an empirical set $\hat{\mathcal{U}}_t$ consisting of the top $\lceil \delta_{prior,t}\, L_{s_T} \rceil$ observations with highest $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$, and estimate the objective as an expectation over that set. This raises two challenges, particularly when considering continuous-valued observations. First, any specific continuous observation $\mathbf{o}_{t+1}$ will almost surely not appear in the set of samples, leading to $P\left[\mathbf{o}_{1:T} \in \hat{\mathcal{U}}_t \mid \mathbf{o}_{0:t+1}\right] = 0$ with probability 1. Second, a specific

continuous observation $\mathbf{o}_t$ that *has* been sampled will almost surely appear only once in the sample set.

Fortunately, rearrangement of (3.9) through Bayes' theorem provides a means to estimate $\delta_{prior,t+1}$ for observations that have not been seen, by using

$$\begin{aligned}
\delta_{prior,t+1} &= \frac{P\left[\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t}, \mathbf{o}_{1:T} \in \hat{\mathcal{U}}_t\right] P\left[\mathbf{o}_{1:T} \in \mathcal{U}_t \mid \mathbf{o}_{0:t}\right]}{P\left[\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t}\right]} \\
&\approx \frac{\hat{p}(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t}, \mathbf{o}_{1:T} \in \hat{\mathcal{U}}_t)\ P\left[\mathbf{o}_{1:T} \in \mathcal{U}_t \mid \mathbf{o}_{0:t}\right]}{\hat{p}(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t})} \\
&= \frac{\hat{p}(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t}, \mathbf{o}_{1:T} \in \hat{\mathcal{U}}_t)}{\hat{p}(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t})} \frac{\lceil \delta_{prior,t}\ L_{s_T} \rceil}{L_{s_T}}
\end{aligned} \tag{3.10}$$

where $\hat{p}$ is computed as an empirical estimator of probability mass or density, as described in Section 3.4. The estimator $\hat{p}(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t}, \mathbf{o}_{1:T} \in \hat{\mathcal{U}}_t)$ is computed from the set of samples in $\hat{\mathcal{U}}_t$, while $\hat{p}(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t})$ is computed from all samples taken at time step t.

3.3.4 Query-Driven Adaptive Sampling Algorithm Description

The procedure for query-driven adaptive sampling performed using online planning is provided in Algorithm 3, including rescaling of query thresholds and prior specializations described in the preceding sections. At each time step t up to the planning horizon, the query used in MCTS is updated to be $\mathcal{Q}_t = \langle f_{\mathcal{Q}}, J_{\mathcal{Q},t}, \Delta_{\mathcal{Q},t} \rangle$, which includes a rescaled prior specialization in the objective $J_{\mathcal{Q},t}$ and a sufficient condition $\Delta_{\mathcal{Q},t}$ using the observations received so far.

In Algorithm 3, planning is performed at steps $t = 0, \ldots, T-1$ within the while loop starting on line 6. Lines 7 through 10 construct $\mathcal{Q}_t$ using $\delta_{prior,t}$ and $\Delta_{\mathcal{Q},t}$, which are initialized for $t = 0$ on lines 2 through 5. The objective estimator to be used in MCTS is then set to estimate $J_{\mathcal{Q},t}$ conditioned on the observations $\mathbf{o}_{0:t}$.

MCTS is performed for a specified amount of time on line 11, and the best action is executed on line 12, resulting in observation of $\mathbf{o}_{t+1}$. The remaining lines are then concerned with deciding whether to end the mission and recomputing the query to be

used for the next step of MCTS. The estimator $\hat{g}(\mathbf{o}_{1:t+1} \mid \mathbf{o}_0)$ is computed and checked against $\Delta_{\mathcal{Q}}$ in lines 13 through 15. If it is found that $\hat{g}(\mathbf{o}_{1:t+1} \mid \mathbf{o}_0) \geq \Delta_{\mathcal{Q}}$, the sufficient condition is satisfied and the mission ends. If the query uses an information objective, the query threshold is recomputed on line 17. Finally, the prior specialization is also rescaled on line 21.

Algorithm 3: Query-Driven Adaptive Sampling

Input : Planning horizon T, planning time τ, original query $\mathcal{Q} = \langle f_{\mathcal{Q}}, J_{\mathcal{Q}}, \Delta_{\mathcal{Q}} \rangle$

1 $t \leftarrow 0$
2 **if** sufficient condition $\Delta_{\mathcal{Q}}$ **then**
3 $\quad \Delta_{\mathcal{Q},0} \leftarrow \Delta_{\mathcal{Q}}$
4 **if** prior specialization δ_{prior} **then**
5 $\quad \delta_{prior,0} \leftarrow \delta_{prior}$
6 **while** $t < T$ **do**
7 $\quad s_t \leftarrow$ state with $\boldsymbol{x}_{0:t}$
8 $\quad J_{\mathcal{Q},t} \leftarrow J_{\mathcal{Q}}$ using prior specialization $\delta_{prior,t}$
9 $\quad \mathcal{Q}_t \leftarrow \langle f_{\mathcal{Q}}, J_{\mathcal{Q},t}, \Delta_{\mathcal{Q},t} \rangle$
10 $\quad$ Set `ObjectiveEstimator` to estimate $J_{\mathcal{Q},t}(\zeta, \mathbf{o}_{t+1:i} \mid \mathbf{o}_{0:t})$
11 $\quad$ `MCTS(`s_t, T, τ, $\mathcal{Q}_t$`)`
12 $\quad$ Execute $\arg\max_a \hat{Q}(s_t, a)$ and observe $\mathbf{o}_{t+1}$
13 $\quad$ Compute $\hat{g}(\mathbf{o}_{1:t+1} \mid \mathbf{o}_0)$
14 $\quad$ **if** $\hat{g}(\mathbf{o}_{1:t+1} \mid \mathbf{o}_0) \geq \Delta_{\mathcal{Q}}$ **then**
15 $\quad\quad$ **return**
16 $\quad$ **if** $J_{\mathcal{Q}}$ is an information objective and sufficient condition $\Delta_{\mathcal{Q}}$ **then**
17 $\quad\quad \Delta_{\mathcal{Q},t+1} \leftarrow \Delta_{\mathcal{Q}} - \hat{g}(\mathbf{o}_{1:t+1} \mid \mathbf{o}_0)$
18 $\quad$ **else if** sufficient condition $\Delta_{\mathcal{Q}}$ **then**
19 $\quad\quad \Delta_{\mathcal{Q},t+1} \leftarrow \Delta_{\mathcal{Q}}$
20 $\quad$ **if** prior specialization $\delta_{prior,t}$ **then**
21 $\quad\quad \delta_{prior,t+1} \leftarrow \frac{\hat{p}(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t}, \mathbf{o}_{1:T} \in \hat{\mathcal{U}}_t)}{\hat{p}(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t})} \frac{\lceil \delta_{prior,t} \ L_{s_T} \rceil}{L_{s_T}}$
22 **end**

3.4 Sample Based Estimators for Query Objectives

We now discuss the use of sample based estimators for computing conditional probabilities and log probability ratios. We then show how these methods may be used to construct the estimators $\hat{J}(s_t, \mathcal{Q})$ and $\hat{g}(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$ used in query-driven adaptive

sampling.

3.4.1 Estimation of Probabilities and Log Probability Ratios

Our objective estimators require estimates for the conditional probability $p(\zeta \mid \mathbf{o}^{(i)}_{1:T}, \mathbf{o}_0)$ and the ratio of log probabilities $\log p(\zeta, \mathbf{o}^{(i)}_{1:T} \mid \mathbf{o}_0)/p(\zeta \mid \mathbf{o}_0)\ p(\mathbf{o}^{(i)}_{1:T} \mid \mathbf{o}_0)$ for an observation $\mathbf{o}^{(i)}_{1:T}$ using a set of L samples $\mathcal{D} = \{(\zeta^{(i)}, \mathbf{o}^{(i)}_{1:T})\}_{i=1}^{L}$ drawn from $p(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0)$. Naturally, the same techniques may be used for conditioning on $\mathbf{o}_{0:t}$ using samples from $p(\zeta, \mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t})$.

Following from work on estimation of probability and mutual information using k-nearest neighbor estimates, we use k-NN and kernel density estimators to produce our estimates. We refer to the estimates of conditional probability as $\hat{p}(\zeta \mid \mathbf{o}^{(i)}_{0:T})$, and we write the estimate for the ratio of log probabilities as $\hat{lr}(\zeta; \mathbf{o}^{(i)}_{1:T} \mid \mathbf{o}_0)$.

Our objective differs somewhat from the typical use of k-NN methods for estimation of probability density and mutual information. Typically, estimation of expectation and mutual information are performed as

$$\mathbb{E}\left[\zeta \mid \mathbf{o}_0\right] \approx \frac{1}{L}\sum_{i=1}^{L} \zeta^{(i)}$$

$$\mathbb{E}\left[lr(\zeta; \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right] \approx \frac{1}{L}\sum_{i=1}^{L} \hat{lr}(\zeta^{(i)}; \mathbf{o}^{(i)}_{1:T} \mid \mathbf{o}_0).$$

In particular, this does not require explicit estimation of $\hat{p}(\zeta \mid \mathbf{o}^{(i)}_{1:T}, \mathbf{o}_0)$ or $\hat{lr}(\zeta, \mathbf{o}^{(i)}_{1:T} \mid \mathbf{o}_0)$ for $\zeta \neq \zeta^{(i)}$. In contrast, certain objectives used in query-driven adaptive sampling require these quantities explicitly. For example, prior specializations are computed using the samples $\mathbf{o}^{(i)}_{1:T}$ with the highest values of $\mathbb{E}_{\zeta|\mathbf{o}^{(i)}_{1:T},\mathbf{o}_0}\left[\zeta \mid \mathbf{o}^{(i)}_{1:T}, \mathbf{o}_0\right]$ or $\mathbb{E}_{\zeta|\mathbf{o}^{(i)}_{1:T},\mathbf{o}_0}\left[lr(\zeta; \mathbf{o}^{(i)}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}^{(i)}_{1:T}, \mathbf{o}_0\right]$. These expressions do require estimation for $\zeta \neq \zeta^{(i)}$, and this estimation occur even if there is only a single sample of the specific observation $\mathbf{o}^{(i)}_{1:T}$. In this way, our estimators also differ from KL-divergence estimators [20, 107], which could estimate $\mathbb{E}_{\zeta|\mathbf{o}^{(i)}_{1:T},\mathbf{o}_0}\left[lr(\zeta, \mathbf{o}^{(i)}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}^{(i)}_{1:T}, \mathbf{o}_0\right]$ using multiple samples of ζ from $p(\zeta)$ and $p(\zeta \mid \mathbf{o}^{(i)}_{1:T}, \mathbf{o}_0)$.

We begin by separating each pair $(\zeta^{(i)}, \mathbf{o}^{(i)}_{1:T})$ into their continuous parts $({}_c\zeta^{(i)}, {}_c\mathbf{o}^{(i)}_{1:T})$

and discrete parts $({}_d\zeta^{(i)}, {}_d\mathbf{o}_{1:T}^{(i)})$, so that we can distinguish estimates of probability mass and probability density. Since ζ is scalar, one of ${}_c\zeta^{(i)}$ and ${}_d\zeta^{(i)}$ will be empty, but we maintain both for convenience in notation. We then sort the samples into sets sharing the same discrete elements $\mathcal{D}_\alpha := \{({}_c\zeta^{(i)}, {}_c\mathbf{o}_{1:T}^{(i)}) \mid ({}_d\zeta^{(i)}, {}_d\mathbf{o}_{1:T}^{(i)}) = \alpha\}$. We use L_α to denote the size of $\mathcal{D}_\alpha$. There are now 4 cases to consider.

ζ discrete, $\mathbf{o}_{1:T}$ discrete: This case is handled using plug-in estimators with the frequencies of each of the discrete values, resulting in

$$\hat{p}(\zeta \mid \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) = \frac{L_{\zeta, {}_d\mathbf{o}_{1:T}^{(i)}}}{L_{{}_d\mathbf{o}_{1:T}^{(i)}}} \tag{3.11}$$

$$\begin{aligned} \hat{lr}(\zeta; \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) &= \widehat{\log p}(\zeta, \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) - \widehat{\log p}(\zeta \mid \mathbf{o}_0) - \widehat{\log p}(\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) \\ &= G(L_{\zeta, {}_d\mathbf{o}_{1:T}^{(i)}}, L) - G(L_\zeta, L) - G(L_{{}_d\mathbf{o}_{1:T}^{(i)}}, L). \end{aligned} \tag{3.12}$$

The function G was defined in (2.18).

ζ discrete, $\mathbf{o}_{1:T}$ mixed: We compute the conditional probability and log probability ratio as

$$\begin{aligned} p(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) &= \frac{p({}_c\mathbf{o}_{1:T}^{(i)} \mid \zeta, {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)\ p(\zeta, {}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)}{p({}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)\ p({}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)} \\ lr(\zeta; \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) &= \log \frac{p({}_c\mathbf{o}_{1:T}^{(i)} \mid \zeta, {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)\ p(\zeta, {}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)}{p(\zeta \mid \mathbf{o}_0)\ p({}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)\ p({}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)}. \end{aligned}$$

Let $\varepsilon^{\mathbf{o}_{1:T}^{(i)}}$ be the distance to k-th nearest neighbor of ${}_c\mathbf{o}_{1:T}^{(i)}$ among the set of samples $\{{}_c\mathbf{o}_{1:T}^{(j)}\}$ that satisfy ${}_d\mathbf{o}_{1:T}^{(j)} = {}_d\mathbf{o}_{1:T}^{(i)}$. Consistent with the use of k-NN estimators for points within their sample sets, the neighbors of ${}_c\mathbf{o}_{1:T}^{(i)}$ include ${}_c\mathbf{o}_{1:T}^{(i)}$ itself. Then let $k^{\zeta, \mathbf{o}_{1:T}^{(i)}}$ be the number of samples such that $\left\| {}_c\mathbf{o}_{1:T}^{(i)} - {}_c\mathbf{o}_{1:T}^{(j)} \right\|_\infty < \varepsilon^{\mathbf{o}_{1:T}^{(i)}}$, ${}_d\mathbf{o}_{1:T}^{(j)} = {}_d\mathbf{o}_{1:T}^{(i)}$,

and $\zeta^{(j)} = \zeta$ (including ${}_c\mathbf{o}_{1:T}^{(i)}$ itself). Making use of the following approximations

$$\begin{aligned}
\hat{p}({}_c\mathbf{o}_{1:T}^{(i)} \mid \zeta, {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) &= \frac{k^{\zeta, \mathbf{o}_{1:T}^{(i)}}}{L_{\zeta, {}_d\mathbf{o}_{1:T}^{(i)}}\ V_{{}_c\mathbf{o}_{1:T}}(\varepsilon^{\mathbf{o}_{1:T}^{(i)}})} \\
\widehat{\log p}({}_c\mathbf{o}_{1:T}^{(i)} \mid \zeta, {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) &= \psi(k^{\zeta, \mathbf{o}_{1:T}^{(i)}}) - \psi(L_{\zeta, {}_d\mathbf{o}_{1:T}^{(i)}}) - \log V_{{}_c\mathbf{o}_{1:T}}(\varepsilon^{\mathbf{o}_{1:T}^{(i)}}) \\
\hat{p}(\zeta, {}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) &= \frac{L_{\zeta, {}_d\mathbf{o}_{1:T}^{(i)}}}{L} \\
\widehat{\log p}(\zeta, {}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) &= G(L_{\zeta, {}_d\mathbf{o}_{1:T}^{(i)}}, L) \\
\hat{p}({}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) &= \frac{k-1}{L_{{}_d\mathbf{o}_{1:T}^{(i)}}\ V_{{}_c\mathbf{o}_{1:T}}(\varepsilon^{\mathbf{o}_{1:T}^{(i)}})} \\
\widehat{\log p}({}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) &= \psi(k-1) - \psi(L_{{}_d\mathbf{o}_{1:T}^{(i)}}) - \log V_{{}_c\mathbf{o}_{1:T}}(\varepsilon^{\mathbf{o}_{1:T}^{(i)}}) \\
\hat{p}({}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) &= \frac{L_{{}_d\mathbf{o}_{1:T}^{(i)}}}{L} \\
\widehat{\log p}({}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) &= G(L_{{}_d\mathbf{o}_{1:T}^{(i)}}, L) \\
\widehat{\log p}(\zeta \mid \mathbf{o}_{1:T}) &= G(L_\zeta, L)
\end{aligned}$$

leads to the simple estimators

$$\hat{p}(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) = \frac{k^{\zeta, \mathbf{o}_{1:T}^{(i)}}}{k-1} \tag{3.13}$$

$$\begin{aligned}
\hat{lr}(\zeta; \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) = \psi(k^{\zeta, \mathbf{o}_{1:T}^{(i)}}) + \psi(L_{{}_d\mathbf{o}_{1:T}^{(i)}}) - \psi(L_{\zeta, {}_d\mathbf{o}_{1:T}^{(i)}}) - \psi(k-1) \\
+ G(L_{\zeta, {}_d\mathbf{o}_{1:T}^{(i)}}, L) - G(L_\zeta, L) - G(L_{{}_d\mathbf{o}_{1:T}^{(i)}}, L).
\end{aligned} \tag{3.14}$$

The probability estimator above sums to 1 across ζ, because there are $k-1$ total samples less than $\varepsilon^{\mathbf{o}_{1:T}^{(i)}}$ from ${}_c\mathbf{o}_{1:T}^{(i)}$, when ${}_c\mathbf{o}_{1:T}^{(i)}$ is included.

When $\mathbf{o}_{1:T}^{(i)}$ has no discrete part, we have $L_{\zeta, {}_d\mathbf{o}_{1:T}^{(i)}} = L_\zeta$ and $L_{{}_d\mathbf{o}_{1:T}^{(i)}} = L$, which asymptotically recovers the discrete-continuous mutual information estimator of Ross [112].

ζ continuous, $\mathbf{o}_{1:T}$ discrete: We use a kernel density estimator, using a kernel that counts samples within a fixed distance $\varepsilon^{\zeta^{(i)}}$. We choose $\varepsilon^{\zeta^{(i)}}$ as the distance to the k-th nearest neighbor of $\zeta^{(i)}$ among all samples $\{\zeta^{(j)}\}$, including $\zeta^{(i)}$. Then we

let k_d^ζ be the number of samples such that $\left\|\zeta - \zeta^{(j)}\right\|_\infty < \varepsilon^{\zeta^{(i)}}$, and ${}_d\mathbf{o}_{1:T}^{(j)} = {}_d\mathbf{o}_{1:T}^{(i)}$ (including $\zeta^{(i)}$), and use the estimators

$$\hat{p}(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) = \frac{k_d^\zeta}{L_{{}_d\mathbf{o}_{1:T}^{(i)}} V_\zeta(\varepsilon^{\zeta^{(i)}})} = \frac{k_d^\zeta}{2L_{{}_d\mathbf{o}_{1:T}^{(i)}} \varepsilon^{\zeta^{(i)}}} \tag{3.15}$$

$$\begin{aligned} \hat{lr}(\zeta; \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) &= \widehat{\log p}(\zeta \mid \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) - \widehat{\log p}(\zeta \mid \mathbf{o}_0) \\ &= \psi(k_d^\zeta) + \psi(L) - \psi(k-1) - \psi(L_{{}_d\mathbf{o}_{1:T}^{(i)}}). \end{aligned} \tag{3.16}$$

ζ continuous, $\mathbf{o}_{1:T}$ mixed: We compute the conditional probability and log probability ratio as

$$\begin{aligned} p(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) &= \frac{p(\zeta, {}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)\ p({}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)}{p({}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)\ p({}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)} \\ lr(\zeta; \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) &= \log \frac{p(\zeta, {}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)\ p({}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)}{p(\zeta \mid \mathbf{o}_0)\ p({}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)\ p({}_d\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)}. \end{aligned}$$

As before, we set $\varepsilon^{\mathbf{o}_{1:T}^{(i)}}$ to be the distance to the k-th nearest neighbor of ${}_c\mathbf{o}_{1:T}^{(i)}$ among the set of samples $\{{}_c\mathbf{o}_{1:T}^{(j)}\}$ that satisfy ${}_d\mathbf{o}_{1:T}^{(j)} = {}_d\mathbf{o}_{1:T}^{(i)}$, including ${}_c\mathbf{o}_{1:T}^{(i)}$ itself. Then we let $k_d^{\zeta,\mathbf{o}_{1:T}^{(i)}}$ be the number of samples such that $\left\|(\zeta, {}_c\mathbf{o}_{1:T}^{(i)}) - (\zeta^{(j)}, {}_c\mathbf{o}_{1:T}^{(j)})\right\|_\infty < \varepsilon^{\mathbf{o}_{1:T}^{(i)}}$, ${}_d\mathbf{o}_{1:T}^{(j)} = {}_d\mathbf{o}_{1:T}^{(i)}$ (including $(\zeta^{(i)}, {}_c\mathbf{o}_{1:T}^{(i)})$), and k^ζ be the number of samples such that $\left\|\zeta - \zeta^{(j)}\right\|_\infty < \varepsilon^{\mathbf{o}_{1:T}^{(i)}}$, (including $\zeta^{(i)}$). Making use of the following approximations

$$\begin{aligned} \hat{p}(\zeta, {}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) &= \frac{k_d^{\zeta,\mathbf{o}_{1:T}^{(i)}}}{L_{{}_d\mathbf{o}_{1:T}^{(i)}}\ V_{\zeta,\mathbf{o}_{1:T}}(\varepsilon^{\mathbf{o}_{1:T}^{(i)}})} \\ \widehat{\log p}(\zeta, {}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) &= \psi(k_d^{\zeta,\mathbf{o}_{1:T}^{(i)}}) - \psi(L_{{}_d\mathbf{o}_{1:T}^{(i)}}) - \log V_{\zeta,\mathbf{o}_{1:T}}(\varepsilon^{\mathbf{o}_{1:T}^{(i)}}) \\ \hat{p}({}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) &= \frac{k-1}{L_{{}_d\mathbf{o}_{1:T}^{(i)}}\ V_{\mathbf{o}_{1:T}}(\varepsilon^{\mathbf{o}_{1:T}^{(i)}})} \\ \widehat{\log p}({}_c\mathbf{o}_{1:T}^{(i)} \mid {}_d\mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) &= \psi(k-1) - \psi(L_{{}_d\mathbf{o}_{1:T}^{(i)}}) - \log V_{\mathbf{o}_{1:T}}(\varepsilon^{\mathbf{o}_{1:T}^{(i)}}) \\ \widehat{\log p}(\zeta \mid \mathbf{o}_0) &= \psi(k^\zeta) - \psi(L) - \log V_\zeta(\varepsilon^{\mathbf{o}_{1:T}^{(i)}}) \end{aligned}$$

leads to the estimators

$$\hat{p}(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) = \frac{k_d^{\zeta, \mathbf{o}_{1:T}^{(i)}}}{(k-1)\ V_\zeta(\varepsilon^{\mathbf{o}_{1:T}^{(i)}})} = \frac{k_d^{\zeta, \mathbf{o}_{1:T}^{(i)}}}{2(k-1)\ \varepsilon^{\mathbf{o}_{1:T}^{(i)}}}. \tag{3.17}$$

$$\hat{lr}(\zeta; \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) = \psi(k_d^{\zeta, \mathbf{o}_{1:T}^{(i)}}) + \psi(L) - \psi(k-1) - \psi(k^\zeta). \tag{3.18}$$

(3.15) through (3.18) may be evaluated for any ζ, but a compact representations of $\hat{p}$ and $\hat{lr}$ can be generated by noting that $\hat{p}(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)$ and $\hat{p}(\zeta \mid \mathbf{o}_0)$ are piecewise constant in ζ. Once $\varepsilon^{\mathbf{o}_{1:T}^{(i)}}$, L, and $L_{\ d\mathbf{o}_{1:T}^{(i)}}$ have been computed, the probability density depends only on the number of samples that are within $\varepsilon^{\mathbf{o}_{1:T}^{(i)}}$ of ζ, which only changes at points that are exactly $\varepsilon^{\mathbf{o}_{1:T}^{(i)}}$ from a sample $\zeta^{(j)}$. By considering these points in order, the piecewise constant ranges of the density function can be constructed.

Algorithm 4 describes the procedure for generating the piecewise probability density function in the form of a set triples (l, u, ρ), where $p(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) = \rho$ for $l < \zeta < u$. First, the set of all observed $\zeta^{(j)}$ for which $\left\| {}_c\mathbf{o}_{1:T}^{(i)} - {}_c\mathbf{o}_{1:T}^{(j)} \right\|_\infty < \varepsilon^{\mathbf{o}_{1:T}^{(i)}}$ and ${}_d\mathbf{o}_{1:T}^{(j)} = {}_d\mathbf{o}_{1:T}^{(i)}$ are collected in a set $\hat{\mathcal{Z}}^{(i)}$. A set of lower bounds $lb = \{\zeta^{(j)} - \varepsilon^{\mathbf{o}_{1:T}^{(i)}} \mid \zeta^{(j)} \in \hat{\mathcal{Z}}^{(i)}\}$ and upper bounds $ub = \{\zeta^{(j)} + \varepsilon^{\mathbf{o}_{1:T}^{(i)}} \mid \zeta^{(j)} \in \hat{\mathcal{Z}}^{(i)}\}$ are then constructed, and considered in order from lowest to highest. After passing each lower bound, the density function increases, and after passing each upper bound, the pdf decreases.

Algorithm 5 is very similar, and describes the procedure for generating the piecewise log probability ratio. When $\mathbf{o}_{1:T}$ is mixed, we must consider the full set of ζ samples $\hat{\mathcal{Z}} = \{\zeta^{(j)}\}$ instead of only those for which $\left\| {}_c\mathbf{o}_{1:T}^{(i)} - {}_c\mathbf{o}_{1:T}^{(j)} \right\|_\infty < \varepsilon^{\mathbf{o}_{1:T}^{(i)}}$, because we will need to compute the count k^ζ.

3.4.2 Estimation of Objectives

We now discuss how to generate estimates for each objective from the set of L samples $\mathcal{D}$. As shown in Algorithm 6, each type of objective; value, probability, and information, uses a specialized estimator that will return a list of L values, each representing an estimate for a single sample $\mathbf{o}_{1:T}^{(i)}$. The appropriate specialized estimator is called by `CallSpecializedEstimator`, with specific forms described in the next sections.

Algorithm 4: GeneratePiecewiseDensity

Input : Conditioned observation $\mathbf{o}_{1:T}^{(i)}$, set of samples $\mathcal{D}_{{}_d\mathbf{o}_{1:T}^{(i)}}$, distance $\varepsilon^{\mathbf{o}_{1:T}^{(i)}}$, discrete sample count $L_{{}_d\mathbf{o}_{1:T}^{(i)}}$ (if $\mathbf{o}_{1:T}$ discrete), k (if $\mathbf{o}_{1:T}$ continuous)

Output: Piecewise constant density for $p(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)$ in *pdf*

if $\mathbf{o}_{1:T}$ discrete **then** `// Evaluating (3.15)`
  $den \leftarrow 2L_{{}_d\mathbf{o}_{1:T}^{(i)}}\, \varepsilon^{\mathbf{o}_{1:T}^{(i)}}$
  $\hat{\mathcal{Z}}^{(i)} \leftarrow \{\zeta^{(j)} \mid (\zeta^{(j)}, {}_c\mathbf{o}_{1:T}^{(j)}) \in \mathcal{D}_{{}_d\mathbf{o}_{1:T}^{(i)}}\}$
else `// Evaluating (3.17)`
  $den \leftarrow 2(k-1)\, \varepsilon^{\mathbf{o}_{1:T}^{(i)}}$
  $\hat{\mathcal{Z}}^{(i)} \leftarrow \{\zeta^{(j)} \mid (\zeta^{(j)}, {}_c\mathbf{o}_{1:T}^{(j)}) \in \mathcal{D}_{{}_d\mathbf{o}_{1:T}^{(i)}}, \left\| {}_c\mathbf{o}_{1:T}^{(i)} - {}_c\mathbf{o}_{1:T}^{(j)} \right\|_\infty < \varepsilon^{\mathbf{o}_{1:T}^{(i)}}\}$
$lb \leftarrow \{\zeta^{(j)} - \varepsilon^{\mathbf{o}_{1:T}^{(i)}} \mid \zeta^{(j)} \in \hat{\mathcal{Z}}^{(i)}\}$
$ub \leftarrow \{\zeta^{(j)} + \varepsilon^{\mathbf{o}_{1:T}^{(i)}} \mid \zeta^{(j)} \in \hat{\mathcal{Z}}^{(i)}\}$
$l \leftarrow \min\ lb$, $lb \leftarrow lb \setminus \{l\}$
$pdf \leftarrow [(-\infty, l, 0)]$
$k_d \leftarrow 1$
while ub is not empty **do**
  $u \leftarrow \min\ lb \cup ub$
  $pdf \leftarrow pdf \cup [(l, u, k_d/den)]$
  if $u \in lb$ **then**
    $k_d \leftarrow k_d + 1$
  else
    $k_d \leftarrow k_d - 1$
  end
  remove u from lb or ub
  $l \leftarrow u$
end
$pdf \leftarrow pdf \cup [(l, \infty, 0)]$
return *pdf*

Algorithm 5: GeneratePiecewiseLogRatio

Input : Conditioned observation $\mathbf{o}_{1:T}^{(i)}$, set of samples $\mathcal{D}_{{}_d\mathbf{o}_{1:T}^{(i)}}$, distance $\varepsilon^{\mathbf{o}_{1:T}^{(i)}}$, discrete sample count $L_{{}_d\mathbf{o}_{1:T}^{(i)}}$ (if $\mathbf{o}_{1:T}$ discrete), k (if $\mathbf{o}_{1:T}$ continuous)

Output: Piecewise constant $\hat{lr}(\zeta; \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)$ in lrf

if $\mathbf{o}_{1:T}$ discrete **then** `// Evaluating (3.16)`
  $\hat{\mathcal{Z}}^{(i)} \leftarrow \{\zeta^{(j)} \mid (\zeta^{(j)}, {}_c\mathbf{o}_{1:T}^{(j)}) \in \mathcal{D}_{{}_d\mathbf{o}_{1:T}^{(i)}}\}$
else `// Evaluating (3.18)`
  $\hat{\mathcal{Z}}^{(i)} \leftarrow \{\zeta^{(j)}\}$
$lb \leftarrow \{\zeta^{(j)} - \varepsilon^{\mathbf{o}_{1:T}^{(i)}} \mid \zeta^{(j)} \in \hat{\mathcal{Z}}^{(i)}\}$, $ub \leftarrow \{\zeta^{(j)} + \varepsilon^{\mathbf{o}_{1:T}^{(i)}} \mid \zeta^{(j)} \in \hat{\mathcal{Z}}^{(i)}\}$
$lrf \leftarrow []$, $l \leftarrow -\infty$
$k_d^{\zeta} \leftarrow 0$, $k^{\zeta} \leftarrow 0$, $k_d^{\zeta, \mathbf{o}_{1:T}^{(i)}} \leftarrow 0$
while ub is not empty **do**
  $u \leftarrow \min\ lb \cup ub$
  $\mathbf{o}_{1:T}^{(j)} \leftarrow$ sample such that $\zeta^{(j)} = u$
  if $\mathbf{o}_{1:T}$ discrete **then**
    $lrf \leftarrow lrf \cup [(l, u, \psi(k_d^{\zeta}) + \psi(L) - \psi(k-1) - \psi(L_{{}_d\mathbf{o}_{1:T}^{(i)}}))]$
    if $u \in lb$ **then** $k_d^{\zeta} \leftarrow k_d^{\zeta} + 1$ **else** $k_d^{\zeta} \leftarrow k_d^{\zeta} - 1$
  else
    $lrf \leftarrow lrf \cup [(l, u, \psi(k_d^{\zeta, \mathbf{o}_{1:T}^{(i)}}) + \psi(L) - \psi(k-1) - \psi(k^{\zeta}))]$
    if $u \in lb$ **then**
      $k^{\zeta} \leftarrow k^{\zeta} + 1$
      if $\left\| {}_c\mathbf{o}_{1:T}^{(i)} - {}_c\mathbf{o}_{1:T}^{(j)} \right\|_{\infty} < \varepsilon^{\mathbf{o}_{1:T}^{(i)}}$ and ${}_d\mathbf{o}_{1:T}^{(i)} = {}_d\mathbf{o}_{1:T}^{(j)}$ **then**
        $k_d^{\zeta, \mathbf{o}_{1:T}^{(i)}} \leftarrow k_d^{\zeta, \mathbf{o}_{1:T}^{(i)}} + 1$
    else
      $k^{\zeta} \leftarrow k^{\zeta} - 1$
      if $\left\| {}_c\mathbf{o}_{1:T}^{(i)} - {}_c\mathbf{o}_{1:T}^{(j)} \right\|_{\infty} < \varepsilon^{\mathbf{o}_{1:T}^{(i)}}$ and ${}_d\mathbf{o}_{1:T}^{(i)} = {}_d\mathbf{o}_{1:T}^{(j)}$ **then**
        $k_d^{\zeta, \mathbf{o}_{1:T}^{(i)}} \leftarrow k_d^{\zeta, \mathbf{o}_{1:T}^{(i)}} - 1$
  remove u from lb or ub
  $l \leftarrow u$
end
return lrf

When a prior and/or posterior specialization is given, the specialized estimator will be an estimate for $g(\mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)$, while in other cases it will be sufficient to return an estimate for $f(\zeta, \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)$. The value returned from Algorithm 6 is the expectation over the top δ_{prior} percentile of these values, computed as the average over the top $\lceil \delta_{prior}\ L \rceil$ elements computed. No prior specialization is equal to the case where $\delta_{prior} = 1$. When the samples are computed a state s_t, the final value returned is the desired estimator $\hat{J}(s_t, \mathcal{Q})$.

Algorithm 6: ObjectiveEstimator

Input : Objective to estimate $J_{\mathcal{Q}}$, set of samples $\mathcal{D}$, number of samples L
Output: Objective estimate $\hat{J}$

$\delta_{prior} \leftarrow$ prior specialization in $J_{\mathcal{Q}}$
$est \leftarrow$ `CallSpecializedEstimator(`$\mathcal{D}$`)`
$sorted \leftarrow$ `SortDecreasing(`est`)`
$\hat{J} \leftarrow 0$
for $i = 1, \ldots, \lceil \delta_{prior}\ L \rceil$
  $\hat{J} \leftarrow \hat{J} + sorted_i$
return $\hat{J} / \lceil \delta_{prior}\ L \rceil$

The following sections describe the estimators that can be called as part of the routine `CallSpecializedEstimator`.

3.4.3 Estimation of Value Objectives

Without posterior specialization, a value objective returns the expected value of the query output, $J_{\mathcal{Q}}(\zeta, \mathbf{o}_{1:T}) = \mathbb{E}[\zeta \mid \mathbf{o}_0]$. Without a prior specialization, it is well known that this values can be estimated simply as the empirical average of samples $\{\zeta^{(j)}\}$, so it is sufficient to return those values for use in the objective estimator directly.

However, when a prior specialization is provided, it is no longer sufficient to return these values, because the average of the top δ_{prior} percentile of samples $\{\zeta^{(j)}\}$ will typically exceed the average of the top δ_{prior} percentile of samples $\{g(\mathbf{o}_{1:T}^{(j)} \mid \mathbf{o}_0)\}$. It is therefore necessary to compute estimates for $\mathbb{E}_{\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0}\left[\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0\right]$, which is done by estimating $p(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)$ for each $\mathbf{o}_{1:T}^{(i)}$, and computing the sum or integral over all ζ. In the case of continuous ζ, the integral of the piecewise constant *pdf* can be

computed exactly by considering each interval. This procedure is in Algorithm 7.

Algorithm 7: ValueEstimator

Input : Set of samples $\mathcal{D}$
Output: Appropriate set *est* to be averaged for value objective estimation

if $\delta_{prior} = 1$ **then**
  return $\left\{\zeta^{(i)} \mid (\zeta^{(i)}, \mathbf{o}_{1:T}^{(i)}) \in \mathcal{D}\right\}$
else if ζ discrete **then**
  return $\left\{\sum_\zeta \zeta\ \hat{p}(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) \mid (\zeta^{(i)}, \mathbf{o}_{1:T}^{(i)}) \in \mathcal{D}\right\}$
else
  $est \leftarrow \{\}$
  for $(\zeta^{(i)}, \mathbf{o}_{1:T}^{(i)}) \in \mathcal{D}$
    $pdf \leftarrow$ `GeneratePiecewiseDensity(`$\mathbf{o}_{1:T}^{(i)}, \mathcal{D}_{d\mathbf{o}_{1:T}^{(i)}}, \varepsilon^{\mathbf{o}_{1:T}^{(i)}}, L_{d\mathbf{o}_{1:T}^{(i)}}, k$`)`
    $est \leftarrow est \cup \left\{\int \zeta\ pdf\ d\zeta\right\}$
  return *est*

The first form of posterior specialization permitted in a value objective is a constant posterior specialization, so that $J(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[P\left[\zeta \in \mathcal{V}_{const} \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_0\right]$. The specialized estimator in Algorithm 8 approximates $P\left[\zeta \in \mathcal{V}_{const} \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0\right]$ for each $\mathbf{o}_{1:T}^{(i)}$, using either the discrete probability estimator directly or integral of the piecewise probability density over $\mathcal{V}_{const}$.

Algorithm 8: ConstantPosteriorSpecializationValueEstimator

Input : Set of samples $\mathcal{D}$, domain $\mathcal{V}_{const}$
Output: Estimates of $P\left[\zeta \in \mathcal{V}_{const} \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0\right]$ in *est*

$est \leftarrow \{\}$
for $(\zeta^{(i)}, \mathbf{o}_{1:T}^{(i)}) \in \mathcal{D}$
  if ζ discrete **then**
    $est \leftarrow est \cup \left\{\sum_{\zeta \in \mathcal{V}_{const}} \hat{p}(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_{1:T})\right\}$
  else
    $pdf \leftarrow$ `GeneratePiecewiseDensity(`$\mathbf{o}_{1:T}^{(i)}, \mathcal{D}_{d\mathbf{o}_{1:T}^{(i)}}, \varepsilon^{\mathbf{o}_{1:T}^{(i)}}, L_{d\mathbf{o}_{1:T}^{(i)}}, k$`)`
    $est \leftarrow est \cup \left\{\int_{\mathcal{V}_{const}} pdf\ d\zeta\right\}$
return *est*

The second form of posterior specialization permitted in a value objective is a max value posterior specialization, so that $J(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[P\left[\zeta \in \mathcal{V}_S^* \mid \mathbf{o}_{0:T}\right] \mid \mathbf{o}_0\right]$ where $\mathcal{V}_S^* = \arg\max_{\mathcal{V} \subseteq \mathcal{Z}^+(\mathbf{o}_{0:T}), |\mathcal{V}| \leq S} \sum_{\zeta \in \mathcal{V}} \zeta$. The specialized estimator in Algorithm

9 approximates $P\left[\zeta \in \mathcal{V}_S^* \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0\right]$ for each $\mathbf{o}_{1:T}^{(i)}$, using either the discrete probability estimator on the largest ζ with positive probability, or the integral of the piecewise probability density over the largest ζ with non-zero density. In the continuous ζ case, the max cardinality S may be reached in the middle of a segment of piecewise constant density.

Algorithm 9: MaximumPosteriorSpecializationValueEstimator

Input : Set of samples $\mathcal{D}$, size of domain S
Output: Estimates of $P\left[\zeta \in \mathcal{V}_S^* \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0\right]$ in *est*

$est \leftarrow \{\}$
for $(\zeta^{(i)}, \mathbf{o}_{1:T}^{(i)}) \in \mathcal{D}$
  $\hat{P} \leftarrow 0$, $S_{used} \leftarrow 0$
  if ζ discrete **then**
    for $\zeta \in$ `SortDecreasing(`$\mathcal{Z}$`)`
      if $\hat{p}(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) > 0$ **then**
        $\hat{P} \leftarrow \hat{P} + \hat{p}\left[\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0\right]$
        $S_{used} \leftarrow S_{used} + 1$
        if $S_{used} = S$ **then**
          $est \leftarrow est \cup \left\{\hat{P}\right\}$
          break
  else
    $pdf \leftarrow$ `GeneratePiecewiseDensity(`$\mathbf{o}_{1:T}^{(i)}, \mathcal{D}_{{}_d\mathbf{o}_{1:T}^{(i)}}, \varepsilon^{\mathbf{o}_{1:T}^{(i)}}, L_{{}_d\mathbf{o}_{1:T}^{(i)}}, k$`)`
    for $(l, u, \rho) \in$ `SortDecreasingInU(`pdf`)`
      if $\rho > 0$ **then**
        $\hat{P} \leftarrow \hat{P} + \rho \ \min(u - l, S - S_{used})$
        $S_{used} \leftarrow S_{used} + \min(u - l, S - S_{used})$
        if $S_{used} = S$ **then**
          $est \leftarrow est \cup \left\{\hat{P}\right\}$
          break
return est

3.4.4 Estimation of Probability Objectives

Without posterior specialization, a probability objective returns the expected conditional probability, $J(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}\left[p(\zeta \mid \mathbf{o}_{0:T}) \mid \mathbf{o}_0\right]$. Just as in the value estimator case, when no prior specialization is provided, this can be estimated as the empirical

average of $\left\{\hat{p}(\zeta^{(i)} \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0)\right\}$, but when a prior specialization is provided we must estimate $\mathbb{E}_{\zeta|\mathbf{o}_{1:T}^{(i)},\mathbf{o}_0}\left[p(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0\right]$ explicitly. This procedure is described in Algorithm 10.

Algorithm 10: ProbabilityEstimator

Input : Set of samples $\mathcal{D}$
Output: Appropriate set *est* to be averaged for probability objective estimation

```
if δ_prior = 1 then
    return {p̂(ζ^(i) | o_{1:T}^(i), o_0) | (ζ^(i), o_{1:T}^(i)) ∈ D}
else if ζ discrete then
    return {Σ_ζ p̂(ζ | o_{1:T}^(i), o_0)^2 | (ζ^(i), o_{1:T}^(i)) ∈ D}
else
    est ← {}
    for (ζ^(i), o_{1:T}^(i)) ∈ D
        pdf ← GeneratePiecewiseDensity(o_{1:T}^(i), D_{d o_{1:T}^(i)}, ε^{o_{1:T}^(i)}, L_{d o_{1:T}^(i)}, k)
        est ← est ∪ {Σ_{(l,b,ρ)∈pdf} ρ^2 (u − l)}
    return est
```

The only form of posterior specialization permitted in a value objective is a max value posterior specialization, so that $J(\zeta, \mathbf{o}_{1:T}) = \mathbb{E}_{\mathbf{o}_{1:T}}\left[P\left[\zeta \in \mathcal{V}_S^* \mid \mathbf{o}_{1:T}\right]\right]$ where $\mathcal{V}_S^* = \arg\max_{\mathcal{V} \subseteq \mathcal{Z}^+(\mathbf{o}_{1:T}), |\mathcal{V}| \leq S} \sum_{\zeta \in \mathcal{V}} p(\zeta \mid \mathbf{o}_{1:T})$. The specialized estimator in Algorithm 11 approximates and returns $P\left[\zeta \in \mathcal{V}_S^* \mid \mathbf{o}_{1:T}^{(i)}\right]$ for each $\mathbf{o}_{1:T}^{(i)}$ by summing the largest discrete probabilities or regions with the largest probability density.

3.4.5 Estimation of Information Objectives

Finally, information objectives are used without posterior specializations, and the objective returns $J(\zeta, \mathbf{o}_{1:T} \mid \mathbf{o}_0) = \mathbb{E}\left[lr(\zeta; \mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right]$. Like the value and probability objective estimators, when there is no prior specialization, the objective can be estimated as the average of $\left\{\hat{lr}(\zeta^{(i)}; \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0)\right\}$, while with a prior specialization we must estimate $\mathbb{E}_{\zeta|\mathbf{o}_{1:T}^{(i)},\mathbf{o}_0}\left[lr(\zeta; \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0\right]$ explicitly. This procedure is described in Algorithm 12.

Algorithm 11: MaximumPosteriorSpecializationProbabilityEstimator

Input : Set of samples $\mathcal{D}$, size of domain S

Output: Estimates of $P\left[\zeta \in \mathcal{V}_S^* \mid \mathbf{o}_{1:T}^{(i)}\right]$ in est

$est \leftarrow \{\}$
for $(\zeta^{(i)}, \mathbf{o}_{1:T}^{(i)}) \in \mathcal{D}$
  $\hat{P} \leftarrow 0$, $S_{used} \leftarrow 0$
  if ζ discrete **then**
    $pmf \leftarrow \{\hat{p}(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) \mid \zeta \in \mathcal{Z}\}$
    for $\hat{p} \in$ `SortDecreasingInP(`pmf`)`
      $\hat{P} \leftarrow \hat{P} + \hat{p}$
      $S_{used} \leftarrow S_{used} + 1$
      if $S_{used} = S$ **then**
        $est \leftarrow est \cup \left\{\hat{P}\right\}$
        break
  else
    $pdf \leftarrow$ `GeneratePiecewiseDensity(`$\mathbf{o}_{1:T}^{(i)}, \mathcal{D}_{{}_d\mathbf{o}_{1:T}^{(i)}}, \varepsilon^{\mathbf{o}_{1:T}^{(i)}}, L_{{}_d\mathbf{o}_{1:T}^{(i)}}, k$`)`
    for $(l, u, \rho) \in$ `SortDecreasingInRho(`pdf`)`
      $\hat{P} \leftarrow \hat{P} + \rho \ \min(u - l, S - S_{used})$
      $S_{used} \leftarrow S_{used} + \min(u - l, S - S_{used})$
      if $S_{used} = S$ **then**
        $est \leftarrow est \cup \{\hat{P}\}$
        break
return est

Algorithm 12: InformationEstimator

Input : Set of samples $\mathcal{D}$

Output: Appropriate set est to be averaged for information estimation

if $\delta_{prior} = 1$ **then**
  return $\left\{\hat{lr}(\zeta^{(i)}; \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) \mid (\zeta^{(i)}, \mathbf{o}_{1:T}^{(i)}) \in \mathcal{D}\right\}$
else if ζ discrete **then**
  return $\left\{\sum_\zeta \hat{p}(\zeta \mid \mathbf{o}_{1:T}^{(i)}, \mathbf{o}_0) \ \hat{lr}(\zeta^{(i)}; \mathbf{o}_{1:T}^{(i)} \mid \mathbf{o}_0) \mid (\zeta^{(i)}, \mathbf{o}_{1:T}^{(i)}) \in \mathcal{D}\right\}$
else
  $est \leftarrow \{\}$
  for $(\zeta^{(i)}, \mathbf{o}_{1:T}^{(i)}) \in \mathcal{D}$
    $pdf \leftarrow$ `GeneratePiecewiseDensity(`$\mathbf{o}_{1:T}^{(i)}, \mathcal{D}_{{}_d\mathbf{o}_{1:T}^{(i)}}, \varepsilon^{\mathbf{o}_{1:T}^{(i)}}, L_{{}_d\mathbf{o}_{1:T}^{(i)}}, k$`)`
    $plr \leftarrow$ `GeneratePiecewiseLogRatio(`$\mathbf{o}_{1:T}^{(i)}, \mathcal{D}_{{}_d\mathbf{o}_{1:T}^{(i)}}, \varepsilon^{\mathbf{o}_{1:T}^{(i)}}, L_{{}_d\mathbf{o}_{1:T}^{(i)}}, k$`)`
    $est \leftarrow est \cup \left\{\sum_{(l,u,\rho) \in pdf, (l,u,lr) \in plr} \rho \ lr \ (u - l)\right\}$
  return est

3.4.6 Estimation of Sufficient Condition Satisfaction

After having planned from state s_{t-1} with a sufficient condition, executed action a_{t-1}, and observed $\mathbf{o}_t$, the samples in previous search trees can frequently be used to compute the estimator $\hat{g}(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$. The state s_t in the search tree that follows action a_{t-1} already contains samples which, when calling `CallSpecializedEstimator` using a prior specialization, are used to construct estimates $\hat{g}(\mathbf{o}_t^{(i)} \mid \mathbf{o}_{0:t-1})$ for each $\mathbf{o}_t^{(i)}$. These same estimators can be used to estimate $\hat{g}(\mathbf{o}_t \mid \mathbf{o}_{0:t-1})$, even when $\mathbf{o}_t \notin \{\mathbf{o}_t^{(i)}\}$. For value and probability objectives, this is sufficient to generate $\hat{g}(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$, because $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0) = g(\mathbf{o}_t \mid \mathbf{o}_{0:t-1})$.

In the case of an information objective, $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0) \neq g(\mathbf{o}_t \mid \mathbf{o}_{0:t-1})$, and it is the first value that we actually need. It may be true that a state following $a_{0:t-1}$ already exists in the search tree that was constructed when planning from s_0, but this will not be true in general. Instead, we cache the original environment model trained with observations $\mathbf{o}_0$. After executing action a_{t-1}, we draw samples $\left\{(\zeta^{(i)}, \mathbf{o}_{1:t}^{(i)})\right\}$, and use those to compute $\hat{g}(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$ using the routines described in `InformationEstimator` with a prior specialization.

3.5 Experiments

In this section, we test query-driven adaptive sampling on scenarios inspired by searches for hydrocarbon seeps on the ocean floor, and emergency response to a wildfire. We compare query-driven adaptive sampling against an information-maximizing adaptive sampling approach that maximizes the mutual information between observations and all environment variables.

3.5.1 Hydrocarbon Seep Search Scenario

In our first experiments, we apply query-driven adaptive sampling to the search for hydrocarbon seeps, which is one of the major domains that inspired its development. The scenario includes an remotely operated underwater vehicle exploring a

multi-attribute environment, with queries depending on data outside the environment model. This scenario is modeled after a field deployment described by Vrolijk et al. [144].

In this scenario, we use data of possible seep sites in the Costa Rica continental margin collected by Sahling et al. [114]. Each site is described by its longitude, latitude, and depth, and is given labels of whether there is a seep present, a mound present, a pockmark present, or elevated backscatter at the location. Each of these four labeled attributes are modeled as binary attributes in the environment model. When the remotely operated vehicle visits a site, it observes the unknown attributes at that location.

In practical missions, the presence of mounds and pockmarks and backscatter characteristics of the ocean floor can be observed using long distance sonar before the deployment of a remotely operated vehicle. This means that usually only the presence or absence of seeps at any location is unknown, while the presence of mounds, pockmarks, and backscatter are known. In addition, the probability of seepage at a location where there is no bathymetric evidence is typically considered to be negligible. As such, only sites with one or more mounds, pockmarks, or elevated backscatter are considered as candidate observation sites.

Seep Depth Range Experiment

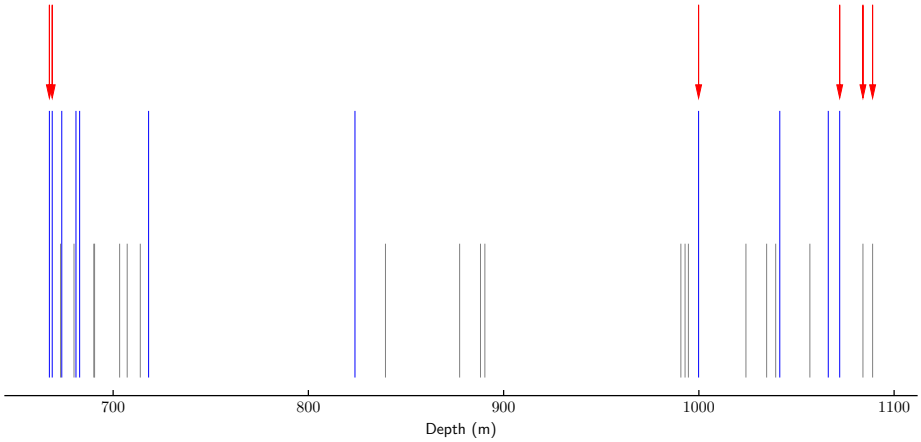
In this experiment, a remotely operated vehicle (ROV) is tasked with determining the depth range at which seeps exist in the environment. That is, the difference in depth between the deepest location there is a seep and the shallowest location there is a seep. We use a finite number of 46 candidate observation sites, so the depth range is a discrete random variable.

We use a query with an information objective, to be optimized over a fixed mission length of 8 observations. The model is trained on full observations at 10 sites, and observations of mounds, pockmarks, and backscatter at an additional 36 locations. We use a fixed structural model, with edges from each of mounds, pockmarks, and backscatter to seeps. In this experiment, we model that the ROV will be lowered

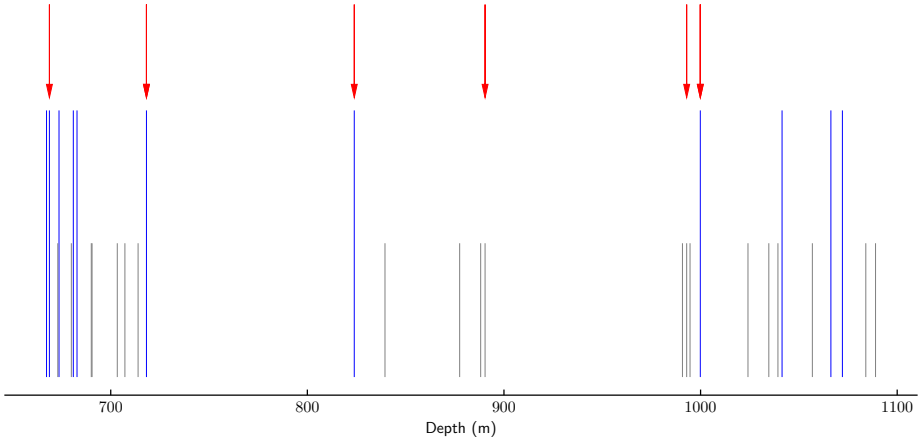
and surfaced for every observation, so there are no dynamics constraints that limit which sites can be explored. We allow 120 seconds of planning time between actions. For comparison, the experiment is performed 10 times with query-driven adaptive sampling and adaptive sampling that maximizes mutual information between observations and the distribution of seeps across the entire environment, using the same Monte Carlo tree search procedure. The observations at the visited locations are taken from the true dataset, rather than simulated.

Results that compare the sites visited by query-driven adaptive sampling and information-maximizing adaptive sampling on one of the 10 missions are given in Figure 3-3, which plots the depth distribution of seeps (blue lines), non-seeps (gray lines), and shows the depths of the locations observed (red arrows) by missions by query-driven adaptive sampling and information-maximizing adaptive sampling. As can be seen in Figure 3-3a, query-driven adaptive sampling focuses observations on deep and shallow sites. Rather than starting by selecting the deepest and shallowest sites, query-driven adaptive sampling first selects a site at approximately 1000 m depth, which is believed very likely to have seepage and significantly reduces uncertainty in the depth range. After seepage at that site is confirmed, query-driven adaptive sampling focuses on deeper and shallower sites. Within 6 observations, it confirms that the shallowest site does have a seep, and that there is not a seep deeper than 1072 m, meaning the depth range of seeps is known exactly. In contrast, the behavior of information-maximizing adaptive sampling in Figure 3-3b shows that most information on seep distribution is gathered over a range of depths, but this is not most beneficial for determining the depth range of seeps. Non-zero entropy in the result exists after 8 observations are taken in the mission. 6 depths are shown in Figure 3-3b because two seeps at depth 718 m and two seeps at depth 1000 m are observed.

Figure 3-4 shows decreases in entropy of seep depth range, averaged across the 10 missions. Here, uncertainty in entropy results from difference in behavior between different random seeds, as well as the fact that environment models trained from different seeds can differ slightly in their predictions. It should also be noted that



(a) Observations selected by query-driven adaptive sampling.



(b) Observations selected by information maximizing adaptive sampling.

Figure 3-3: Depths of observations selected by query-driven adaptive sampling and information maximizing adaptive sampling. Blue lines indicate seeps, gray lines indicate non-seeps, and red arrows indicate sites selected for observation.

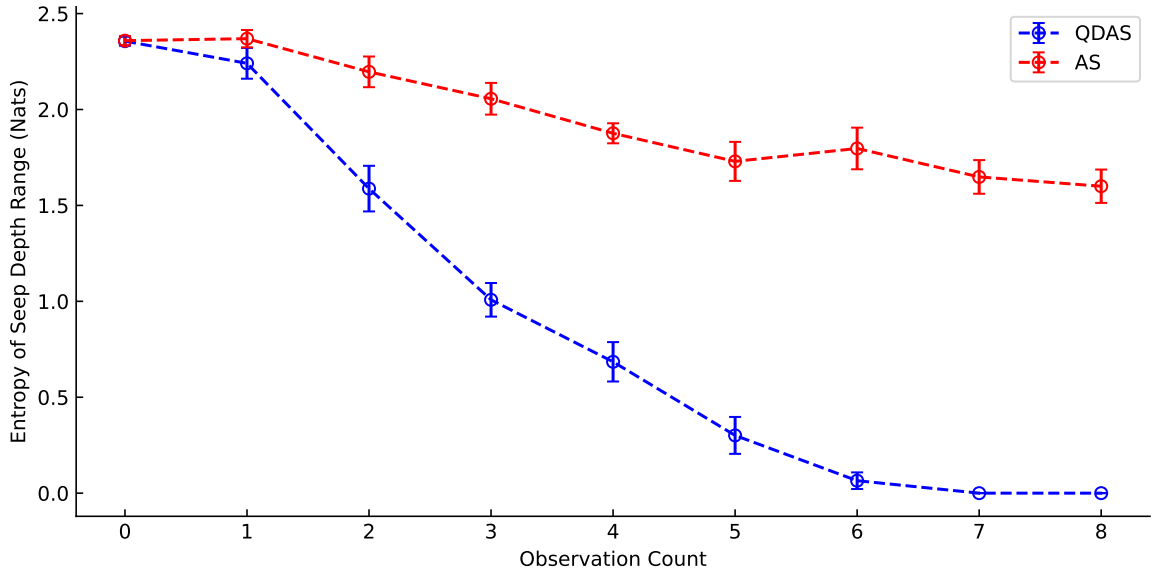


Figure 3-4: Average entropy in seep depth after observations in simulated mission comparing query-driven adaptive sampling (QDAS) and information-maximizing adaptive sampling (AS).

the outcomes of missions are determined from the data set and not drawn from the modeled distribution, meaning that entropy can increase after an observation is taken. The results show a significant reduction in depth range entropy using query-driven adaptive sampling, and there is always no uncertainty in seep depth range after no more than 7 observations. Information-maximizing adaptive sampling, on the other hand, never resolves the question of seep depth, since it does not focus observations on the shallowest and deepest sites.

Seep Search Without Observation Noise

In this experiment, a remotely operated vehicle is tasked with minimizing the number of observations required to find 3 additional seeps in the top 90% of mission outcomes. This experiment demonstrates the capability of query-driven adaptive sampling to identify undesirable outcomes in the bottom 10% of possible observations, and immediately end the mission in those cases.

This mission is modeled using a value query objective, a sufficient condition, and a prior specialization with $\delta_{prior} = 0.9$. The ROV is assumed to be able to

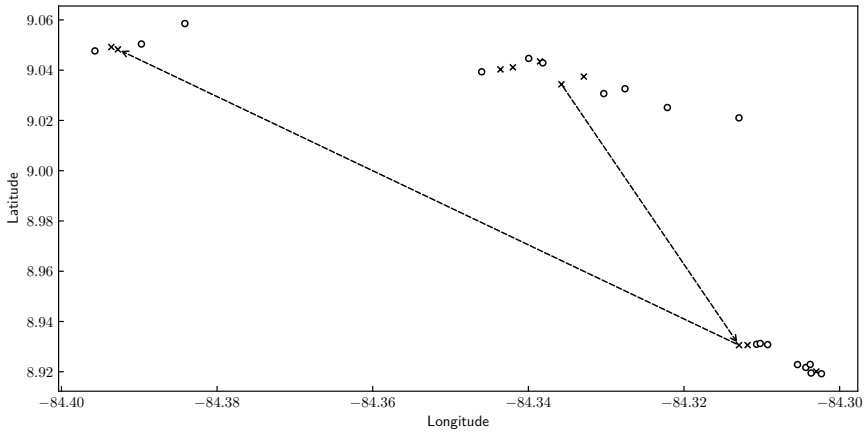
determine exactly whether seeps are present or absent once it has visited a site. The environment model is trained on full observations at 5 sites, and observations of mounds, pockmarks, and backscatter are provided at an additional 28 locations. The training set contains 2 known seeps, and 3 additional seeps are required to be found from among the 28 sites with unobserved seep presence, for a total of 5 known seeps.

In this example, we observe that the 28 candidate sites naturally form 4 separate clusters. We constrain the agent dynamics so that after it leaves a cluster, it cannot return to that same cluster, in order to avoid needlessly complex paths. We allow 300 second of planning time between actions, and repeat the experiment 10 times using query-driven adaptive sampling.

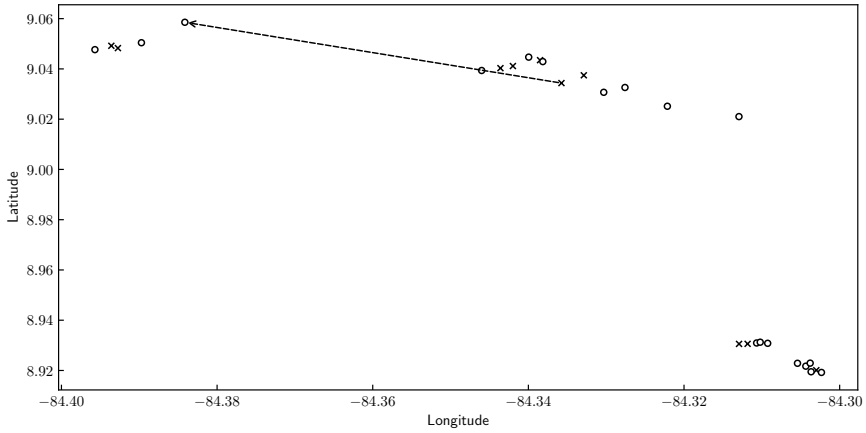
It turns out that the environment model predicts the presence of seeps at certain locations with high confidence, even with very limited data. This occurs because both seeps in the training set occur with elevated backscatter but no mounds and pockmarks, which is not shared by any of the non-seep sites in the training set. The model associates this pattern with seepage, and the pattern is actually a strong indicator of seepage in the dataset, with 8 of 11 sites with backscatter and no mounds or pockmarks being seeps. This means that missions typically proceed by moving to three sites of seepage, as in Figure 3-5a, and immediately ending.

However, behavior when a non-seep site is reached is interesting from the perspective of application of prior specializations. In Figure 3-5b, the environmental model results in a high confidence that the second site visited is a seep, so that the top 90% of outcomes all consider a seep at that site. When the site visited is revealed to not be a seep, query-driven adaptive sampling determines that the realization of the environment is drawn from the 10% of outcomes that are not counted towards the objective. As a result, $\delta_{prior,2}$ is computed to be zero, and the mission immediately ends without satisfaction of the sufficient condition.

In total, the average mission length across the 10 runs is 2.40 ± 0.22, not counting the observations needed to collect the prior data, with every mission that achieves the sufficient condition ending in exactly 3 observations.



(a) Mission that ends with satisfaction of the sufficient condition.



(b) Mission that ends due to prior specialization.

Figure 3-5: Paths generated in the seep search experiment without observation noise. Crosses indicate sites of seepage, circles indicate sites without seepage. Axes are not scaled geographically.

Seep Search With Observation Noise

The previous experiment is simple because the certain seeps in the environment can be predicted with high confidence. To add additional complexity, we repeat the previous experiment, simulating the presence of noise in all observations of seepage, both in the training set and during the mission. This could reflect that the hydrocarbon seep does not constantly release gas, and that the presence of seepage is imprecisely inferred from local evidence. This experiment demonstrates that query-driven adaptive sampling is able to reason about satisfying sufficient conditions involving variables that are never directly observed.

We model that an observation of seepage matches truth with 75% probability, and returns the opposite result with 25% probability. The modeler is aware of this noise, so that there is greater uncertainty in the model predictions. The objective is to minimize the number of observations so that the expected total number of seeps seen, conditioned on the observations and including the training data, exceeds 5 in the top 90% of outcomes. We constrain the agent dynamics as in the previous experiment, and once again allow 300 seconds for planning between observations over 10 runs.

This experiment is interesting because the agent never directly observes seepage. Instead, the truth at the locations visited must be inferred from the noisy observations, and that the top 90% of mission outcomes include negative observations. The most frequent outcome, shown in Figure 3-6a, is once again to visit three seeps, and observe presence of a seep at each. Even though the algorithm is not certain that these locations are seeps, the expected total number of seeps visited exceeds 5, when accounting for the possibility of seeps being at all training locations.

Figure 3-6b shows that having visited 3 additional seeps can be inferred, even when only 2 observations indicate presence of a seep. In this case, an additional 3 seeps are visited after the first three observations of the mission. However, since the second observation is 0, there is not sufficient evidence that five total seeps have been visited until a fourth observation is completed. However, the inferences are not always correct. Figure 3-6c shows a mission in which 2 additional seeps have been

visited, but the expected number visited in total still exceeds 5.

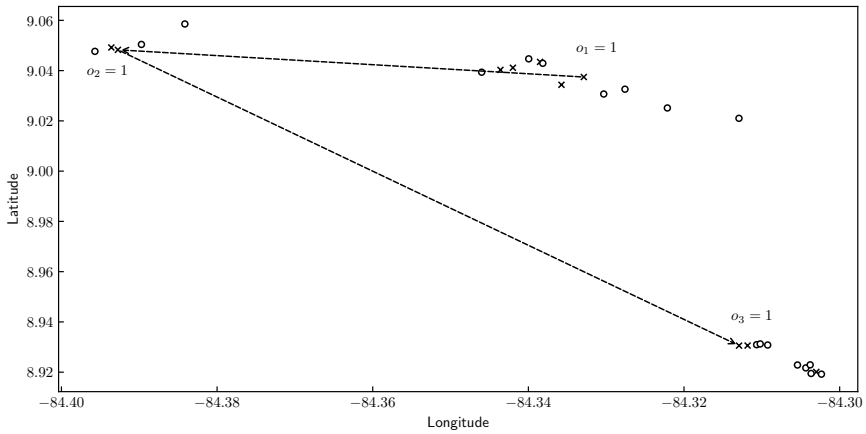
In total, the average mission length across the 10 runs is 4.00 ± 0.45 observations, with an average over those that achieve the sufficient condition of 3.67 ± 0.33 observations.

3.5.2 Fire Escape Scenario

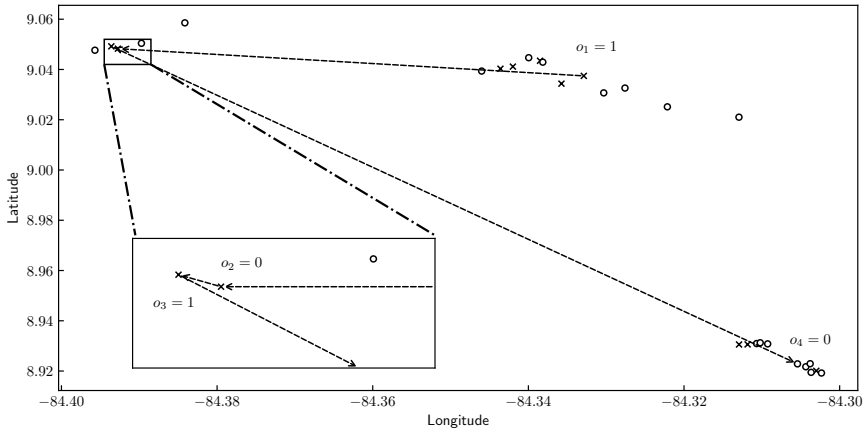
One of the major strengths of query-driven adaptive sampling is that we are able to plan with respect to arbitrary variables of interest, which may potentially be defined by highly complex functions of the environment. The following scenario demonstrates this complexity by performing adaptive sampling in service of identifying escape routes from wildfires and informing emergency responders of where they should focus their efforts. The scenario is loosely inspired by the evacuation of Southern Lake Tahoe.

In this scenario, we consider the road network illustrated in Figure 3-7 in a region subject to a wildfire emergency. Nodes represent major junctions where people can reside, and edges represent roads that allow people to evacuate the area. Table 3.1 gives the modeled coordinates of each node. Each edge is either blocked by fire hazards or unblocked, as modeled by a binary variable located at the center point of each edge (small circles in Figure 3-7). The presence of fire hazards is spatially correlated according to a binary heterogeneous AcyGP model. Nodes A, E, P, and Q are connected to further roads that allow exit from the modeled area. We say that there is an escape route from a node if A, E, P, or Q can be reached from the node by following unblocked roads.

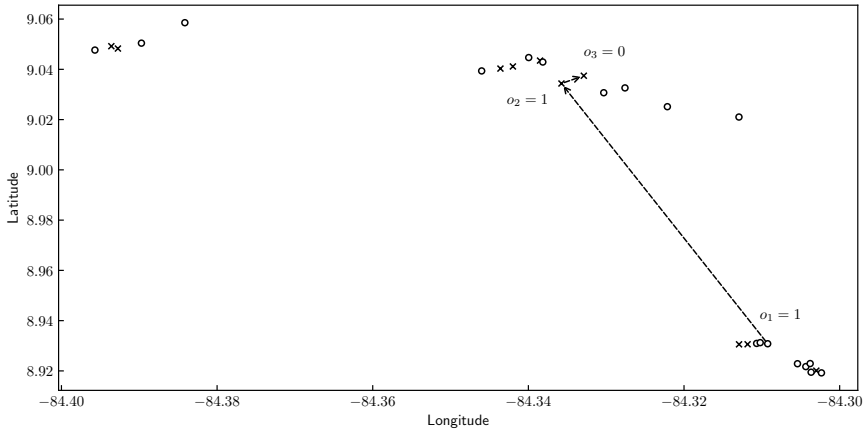
An autonomous drone is dispatched to identify which roads are blocked. A mission is modeled as having time for up to 8 actions. Starting from a specified node, for each action the drone may fly directly to any other node up to 12 units away. If start and end nodes are connected by a road, the drone is assumed to fly above the road and observe if it is blocked with no noise. No meaningful data is taken if the two roads are not connected. To model the dependence of observations on the path taken, $\mathbf{o}_t$ is defined to be 0 if flying over an unblocked road, 1 if flying over a blocked road, and



(a) Mission that ends with satisfaction of the sufficient condition after visiting 3 seeps.



(b) Mission that ends with satisfaction of the sufficient condition after visiting 3 seeps, with only 2 positive observations.



(c) Mission that ends with satisfaction of the sufficient condition without visiting 3 seeps.

Figure 3-6: Paths generated in the seep search experiment with observation noise. Crosses indicate sites of seepage, circles indicate sites without seepage. Axes are not scaled geographically.

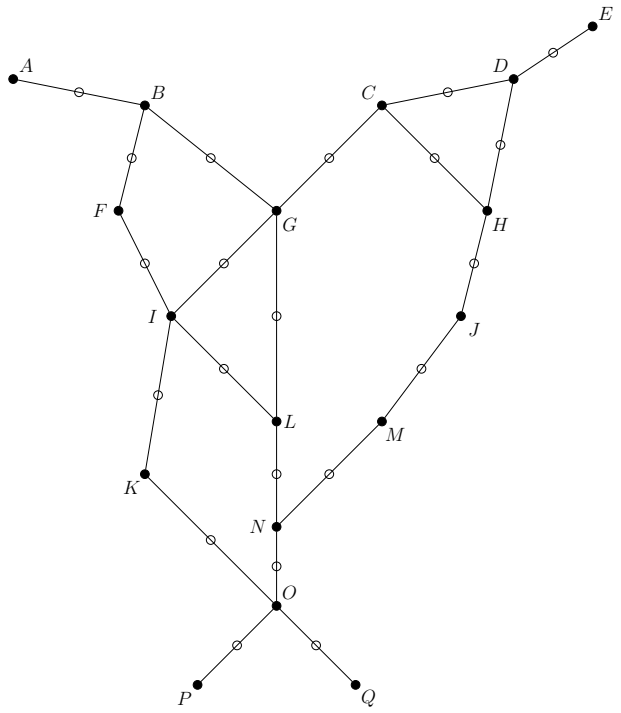


Figure 3-7: Road network used for fire escape scenario experiments.

Node	Coordinates
A	$(0, 0)$
B	$(5, -1)$
C	$(14, -1)$
D	$(19, 0)$
E	$(22, 2)$
F	$(4, -5)$
G	$(10, -5)$
H	$(18, -5)$
I	$(6, -9)$
J	$(17, -9)$
K	$(5, -15)$
L	$(10, -13)$
M	$(14, -13)$
N	$(10, -17)$
O	$(10, -20)$
P	$(7, -23)$
Q	$(13, -23)$

Table 3.1: Road network node coordinates in the fire escape scenario.

a dummy value of 2 if not following a road.

We consider two different experiments in this scenario; one where an escape route must be identified for people within the road network, and another where emergency responders must be decide which roads they should clear. Both experiments show the advantages of using adaptive sampling that is query-focused by focusing observations on roads that are relevant to the query, which outperforms adaptive sampling that maximizes information about the presence of fires throughout the environment. Furthermore, the emergency response experiment shows the utility of posterior specializations.

Confirmed Escape Route Experiment

In this experiment, evacuees at L are searching for an escape route from L that is confirmed to be unblocked. That is, they desire an escape route from L where every road has been observed and found to be unblocked. The objective is to maximize the probability that such an escape route has been found at the conclusion of the mission. Based on the roads traversed in the path $\boldsymbol{x}_{1:T}$, the query function $f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \mathscr{M})$ returns 1 if a confirmed escape route from L has been found, and 0 otherwise. The objective is maximization of value.

At the start of the experiment, we model that roads BF and GI are known to be blocked, while JM, MN, LN, and KO are known to be open. 180 seconds are allowed for planning before each action is taken, and the experiment is repeated 30 times using query-driven adaptive sampling and adaptive sampling maximizing the mutual information between road status and observations. Each repeat uses an environment truth drawn from the Gaussian Process posterior.

Results are given in Table 3.2. Since information-maximizing adaptive sampling attempts to construct a most accurate distribution over the entire map, we never observed it to spatially focus observations and confirm the existence of an escape route. With query-driven adaptive sampling, exactly half of all completed missions resulted in confirmation of an escape route.

AS	QDAS
0.00 ± 0.00	**0.500 ± 0.09**

Table 3.2: Empirical expected success rate in the confirmed escape route experiment for information-maximizing adaptive sampling (AS) and query-driven adaptive sampling (QDAS). Numbers after ± indicate standard errors in the mean.

Emergency Response Experiment

In this experiment, firefighters at node P deploy the drone in order to help them prioritize their resources. After the drone observes $\mathbf{o}_{1:T}$, the firefighters use the data to select a road to clear. The firefighters can only reach a blocked road by following unblocked roads from P. The goal is to use the drone to select the road with the largest probability of creating the most escape routes.

To model this problem, the query function $f_{\mathcal{Q}}(\boldsymbol{x}_{1:T}, \mathscr{M})$ considers all blocked roads reachable by an unblocked path from node P. For each candidate road, it computes the number of nodes in the environment that would have escape routes if the road was cleared. $f_{\mathcal{Q}}$ returns the reachable road that results in the largest number of escape routes when cleared as ζ. Ties are broken deterministically, favoring roads that can be reached following fewer roads from P. Between roads that still tie, a deterministic preference is given that represents secondary preferences such as ease of traversal of the roads and visibility.

The objective in this case is to maximize the expected maximum posterior probability $p(\zeta \mid \mathbf{o}_{1:T})$,

$$\max \ \mathbb{E}\left[\max_{\zeta} \ p(\zeta \mid \mathbf{o}_{1:T})\right].$$

In our framework, this is modeled as a probability objective with a maximum posterior specialization of size 1.

At the start of the experiment, we model that roads BF, CD, FI, GI, HJ, and JM are known to be blocked, while MN, KO, and OP are known to be unblocked. Conditions of all other roads are unknown. 120 seconds are allowed for planning before each action is taken. The experiment is repeated 30 times using query-driven adaptive sampling and adaptive sampling maximizing the mutual information between road

AS	QDAS
0.570 ± 0.038	**0.798 ± 0.029**

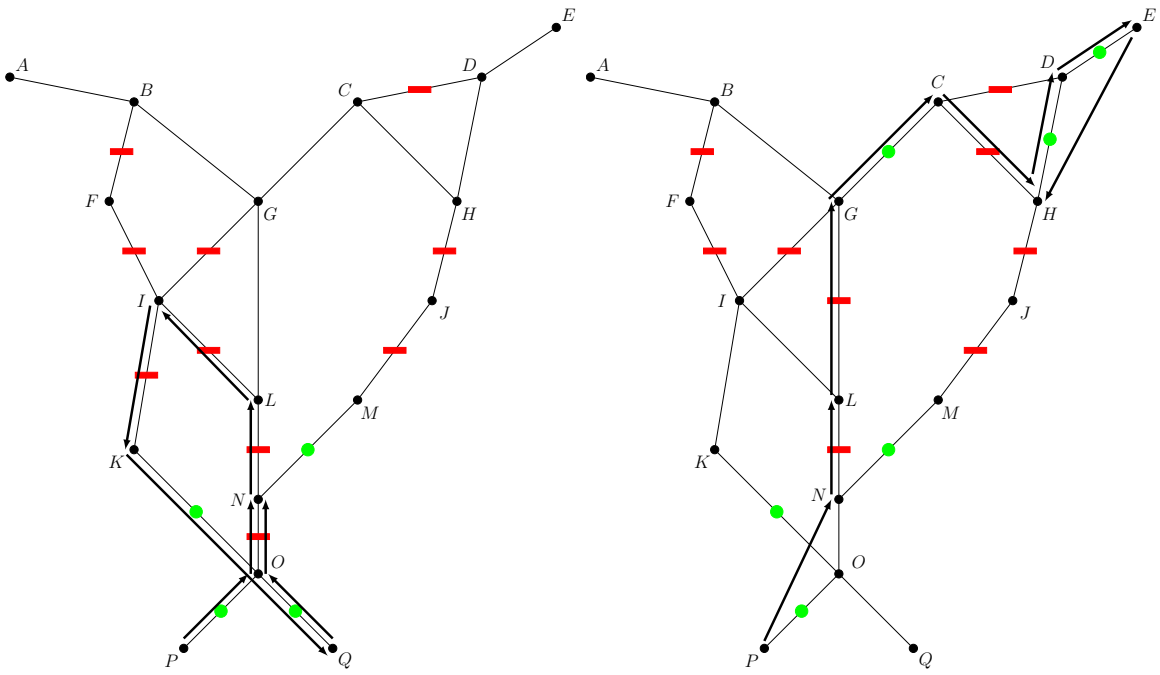
Table 3.3: Empirical expected maximum posterior probabilities in the emergency response experiment for information-maximizing adaptive sampling (AS) and query-driven adaptive sampling (QDAS). Numbers after ± indicate standard errors in the mean.

status and observations. Each repeat uses an environment truth drawn from the Gaussian Process posterior.

Results averaged over 30 experiment runs are given in Table 3.3, where we compare the empirical expected maximum posterior probabilities of query-driven adaptive sampling and information maximizing adaptive sampling. In this experiment, the state of a road is not immediately important unless clearing a second road would open up a new escape route to the first. By using an appropriate objective within query-driven adaptive sampling, the policy is able to focus on paths that are most useful for maximizing posterior probability, rather than maximizing information about the presence of fires in locations that do not direct the responders. This results in a significantly higher posterior probability for query-driven adaptive sampling than information-maximizing adaptive sampling.

Depictions of the differences in behavior between query-driven adaptive sampling and information maximizing adaptive sampling are shown in Figures 3-8 and 3-9, which show the routes taken by query-driven adaptive sampling and information-maximizing adaptive sampling, for two different environment truths. Symbols over the roads indicate the known state of the environment at the conclusion of the mission, either from what was known at the start of the mission, or from observations. Green circles indicate unblocked paths, while red rectangles indicate blocked paths.

Figure 3-8 shows paths taken when fire is relatively widespread. The mission planned by query-driven adaptive sampling in Figure 3-8a identifies that emergency responders cannot travel far from P, and the only choices of routes to clear are IK or NO. Clearing IK will add exactly one additional escape route (from I) and clearing NO will add exactly two additional escape routes (from N and M). The information



(a) Path generated by query-driven adaptive sampling.

(b) Path generated by information maximizing adaptive sampling.

Figure 3-8: Paths generated in the emergency response experiment when the environment has widespread fire hazards, along with the known state of the environment at the conclusion of the mission. Green circles indicate unblocked paths, red rectangles indicate blocked paths. Using query-driven adaptive sampling, all roads reachable by emergency responders are explored, and the best route to clear is known exactly. Information maximizing adaptive sampling leaves uncertainty over which roads are reachable and should be cleared.

gathered by the drone there shows that clearing NO is certainly the best choice, so that $\max_\zeta\ p(\zeta \mid \mathbf{o}_{1:T}) = 1$. Note that this result is actually known by the time the drone reaches node K; the additional actions provide an observation of OQ that does not change the number of nodes with escape routes. The conditions of the top of the map are unknown, but observing those regions does not affect the immediate decision of the emergency responders.

In contrast, the mission generated by information-maximizing adaptive sampling in Figure 3-8b provides more information about the state of the overall environment, but is less useful for directing the emergency responders. After the mission, the statuses of IK and ON are unknown, which means the best route to clear is also unknown. As in the previous case, if NO is blocked, then clearing it will add exactly 2 escape routes. Meanwhile, if IK turns out to be unblocked, then clearing IG could add between zero and three new escape routes (from B, C, and G, depending on the status of AB and BG). But IG cannot be reached if IK is blocked, and clearing and IK will add between 1 and 2 escape routes (from I and L, depending on IL). If the emergency responders deployed and saw ON was blocked, they would not know with certainty whether to clear ON or to travel up OK and clear routes further up that road. Computation of the probabilities in this case reveals that $\max_\zeta\ p(\zeta \mid \mathbf{o}_{1:T}) \approx 0.437$, with the best ζ being ON. Overall, while query-driven adaptive sampling identifies the optimal road to clear, information maximizing adaptive sampling results in significantly lower probability in the best course of action.

Figure 3-9 shows paths taken by query-driven adaptive sampling and information-maximizing adaptive sampling when the fire has not spread as far to the South of the map. When clearing a road results in the same number of escape routes and require the same number of roads from P, our ordering asserts that clearing JM is preferred over GL, which is preferred over GI. After data is gathered by query-driven adaptive sampling in Figure 3-9a, the best path to clear is GL with relatively high probability, because it may generate escape routes from B, C, D, and H in addition to G. Only when both BG and CG are blocked, or when an escape route from G already exists does clearing JM become the best choice. This results in $\max_\zeta\ p(\zeta \mid \mathbf{o}_{1:T}) \approx 0.701$,

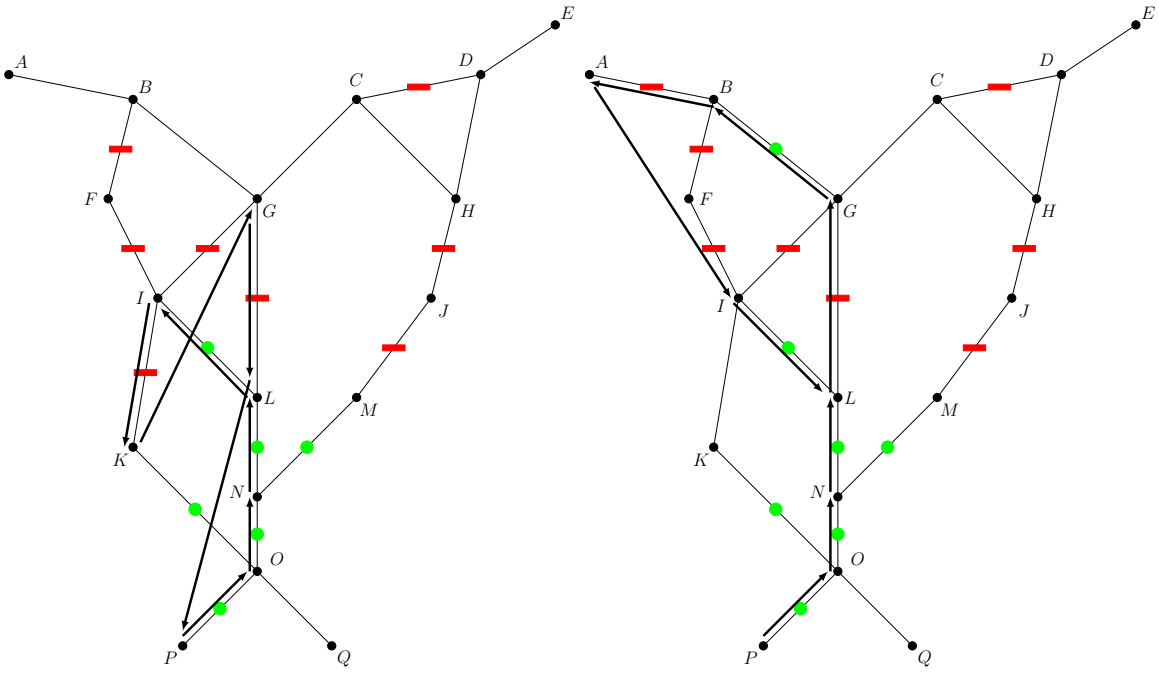
with the best ζ being GL

In contrast, after data is gathered by information-maximizing adaptive sampling in Figure 3-9b, there remains ambiguity about whether I can be reached by following 3 roads, since IK is unobserved. If IK is clear, then clearing GI is usualy preferred. If IK is blocked instead, then GL is preferred. But in both cases, if there already exists an escape route from G, then clearing JM is preferred. The status of IK is quite uncertain, resulting in $\max_\zeta \; p(\zeta \mid \mathbf{o}_{1:T}) \approx 0.469$, with the best ζ being GI. Overall, while neither mission is able to determine the best choice of road to clear with 100% probability, use of query-driven adaptive sampling results in a significantly higher maximum probability for the best known choice.

3.6 Summary

In this chapter, we introduced a method to perform query-directed adaptive sampling using Monte Carlo tree search to produce online unconditional plans. Our MCTS approach embedded complex query estimators within tree search, and supported early termination of rollouts and optimization over a percentile of samples in order to support sufficient conditions and prior specializations in queries. We then discussed how query thresholds and prior specializations must be modified during online planning in order to optimize the query during execution.

Finally, through experiments based on hydrocarbon seep search, and escape from and response to wildfires, we showed that query-driven adaptive sampling outperforms adaptive sampling that maximizes information about all environment variables. By focusing observations to only those that are relevant to the query, questions posed in the forms of queries are able to be answered more precisely using query-driven adaptive sampling.



(a) Path generated by query-driven adaptive sampling.

(b) Path generated by information maximizing adaptive sampling.

Figure 3-9: Paths generated in the emergency response experiment when the environment has fewer fire hazards, along with the known state of the environment at the conclusion of the mission. Green circles indicate unblocked paths, red rectangles indicate blocked paths. Neither approach identifies the best road to clear with 100% probability, but with query-driven adaptive sampling it is recognized that clearing GL is likely to open the most escape routes. With information-maximizing adaptive sampling, the status of IK is unknown, so additional uncertainty over the best route remains because it is unknown whether node I can be reached by traveling over 3 unblocked roads.

Chapter 4

Scalable Monte-Carlo Tree Search with Risk Bounding Functions

In practical exploration missions of novel environments, like in oceanographic campaigns and planetary exploration, an exploring agent is exposed to a significant amount of risk, which may lead to loss of the agent. In exploration problems in particular, it is unclear how much additional information can be gained by taking those risks, so it has been proposed to develop exploration policies subject to a risk bounding function, that constrains permissible risk as a function of reward.

Previous work has explored solving offline for a policy subject to a risk bounding function, but the need to choose actions for every measurement in advance of a mission has two disadvantages. First, the size of the decision tree means that missions that are small enough to be solved are limited in their duration, with typically fewer than 6 actions. Second, an action cannot be generated in advance for every possible outcome in a continuous space. This means approximations are required to estimate reward and risk, but it is difficult to guarantee that risk is bounded as a function of reward when neither are known exactly. In query-driven adaptive sampling, we will require long duration policies that respond to continuous spaces of observations, so in this chapter, we will show that an agent can execute conditional policies that satisfy risk bounding functions by performing online unconditional planning. This will allow us to consider significantly fewer states when planning, and allow us to produce plans

up to 20 actions in length.

Our approach is to apply an extension of Monte Carlo Tree Search (MCTS), that bounds probability of failure, to solve these unconditional plans online in an anytime manner. Our innovation lies in how we define the unconditional planning problems and online planning strategy. We are able to formulate the problems so that the probability of failure constraint is guaranteed to be satisfied in expectation over all possible executions, without introducing large amounts of conservatism. The approach does not need to plan for all measurements explicitly, or constrain planning based only on the measurements that were observed. Through experiments on real bathymetric data and simulated measurements, we show our algorithm allows an agent to take dangerous actions only when the reward justifies the risk. We then verify through Monte Carlo simulations that failure bounds are satisfied.

4.1 Motivation

Scientists commonly perform adaptive sampling missions in order to identify and confirm the existence of high reward regions. In underwater exploration, for example, autonomous vehicles may be tasked with locating regions with high temperatures, algal and plankton blooms, or high concentrations of pollutants or hydrocarbons for the purpose of identifying suitable locations for follow-up studies.

Such missions are performed because the environment is not well understood. New measurements contribute to an improved understanding as they are received, and impact the future of the mission. We require an adaptive system that is capable of updating its plan in response to new measurements to direct it towards high reward locations. The measurements may be drawn from a continuous space, so it is not feasible to assign a unique action to every possible measurement before the mission begins. Furthermore, the approach must be scalable, to allow missions with tens of actions.

However, disturbances and noises acting on an agent make safe autonomous exploration difficult. In underwater applications, the seafloor surface is often known

relatively well, but uncertainty in position accumulates due to unknown currents and inaccuracy in on-board inertial navigation. Navigating close to obstacles incurs some probability of collision, leading to failure, but it is often impossible to conduct a mission without accepting some level of risk. There is therefore a tradeoff between allowed risk and expected reward over which the autonomous system should be able to reason.

In this chapter, we describe a method of planning and executing an adaptive mission that maximizes the expected reward of samples, while limiting the probability of failure of the mission. We define a method of specifying tolerance for failure as a function of reward through a *risk bounding function*, and enforce the *chance constraint* that the expected rate of failure is bounded by the risk bounding function applied to the expected reward over all executions. Our method applies Monte Carlo Tree Search (MCTS) to online unconditional plans, which allows a solution to be found in an anytime manner, making it suitable for on-board autonomy or missions with tight time constraints. Furthermore, we are able to provably enforce the chance constraint without planning for all measurements explicitly, and without limiting allowed probability of failure based only on the observation that were taken.

4.2 Related Work

Adaptive sampling tasks an agent with exploring an environment that is unknown. The environment is either characterized by a known uncertainty field [61, 155], or described by a stochastic process such as a Gaussian process (GP) [16, 77, 90]. It is typical to discretize available actions and perform discrete state space search, though notable exceptions exist, including using Rapidly Exploring Random Trees [61] or genetic algorithms [60].

When tasked with maximizing information measures in a continuous-valued GP, the information depends only on the locations of samples and not their values [16, 76]. It follows that replanning in response to new information is not necessary, and sampling may be directed to regions where the GP is uncertain, but nonetheless

believed to be low. In contrast, missions that require maximization of a function that depends on measurements of the GP, or perform information maximization in heterogeneous environment model, the policy will depend explicitly on observations, and must be adaptive.

Even with discrete action and state spaces, the problem quickly becomes intractable as the search tree branches in both actions and the full history of observations. Fixed horizon planning strategies that consider only the next few actions [77, 93, 126] cannot guarantee chance constraints, as no actions may be possible that satisfy failure bounds late in the policy, and setting a probability threshold for each action can lead to highly suboptimal policies. An entire policy that branches on measurements is found by Low et al. [90], though their experimental results imply small action spaces and relatively few measurements, whereas we consider on the order of 20 actions.

An alternative approach used by Hitz et al. [60] is to plan and begin execution of a full policy that does not depend on measurement outcomes but satisfies a cost constraint. The policy is updated by replanning in response to new measurements. We adopt a similar approach, while the key differences between this work and Hitz et al. in this regard are that our chance constraint applies to the policy as a whole and depends on measurement outcomes.

Chance constrained planning has been examined for motion planning [17, 99], but bounds are applied to non-adaptive plans. These approaches could be used within an online planning framework, and using some of the results in this chapter, can be made to bound probability of failure averaged across all executions. However, the result can be highly suboptimal, and this approach may not even be possible with risk bounding functions for outcomes that repeatedly result in low rewards. RAO* applies chance constraints over policies for Partially Observable Markov Decision Processes [116], but it will search over all possible actions if the heuristic guiding search is poor, and we lack an informative heuristic for adaptive sampling objectives. Our approach differs in that we reason over a more general notion of a chance constraint through a risk bounding function, do not explicitly branch on measurements, and use MCTS

to produce a policy under limited planning time without heuristics.

A different approach for chance constrained planning is to maximize a sum of reward and a weighted penalty for failure in an unconstrained problem. There is no known method of selecting weights to guarantee chance constraint satisfaction, so the unconstrained problem is solved repeatedly with different weights until the solution is observed to satisfy the chance constraint [49]. We consider large problems for which it is intractable to produce a full policy and calculate the probability of failure. Instead, we define sequential approximating problems that guarantee chance constraint satisfaction without needing to explicitly compute reward or probability of failure for the full policy.

4.3 Problem Statement

In this chapter, we consider an agent exploring an environment modeled as a stochastic process. Like in Problem 1, the agent receives observations $\mathbf{o}_t$ at a location $\boldsymbol{x}_t$, which are drawn from a distribution that depends on local environment variables $\mathbf{y}(\boldsymbol{x}_{t+1})$. The agent's objective is to execute a policy that optimizes the expectation of a function $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$. The form of $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$ is identical to that of the query objectives that appear in Chapter 2, in that it is an expectation over a variable of interest determined from a query function..

Separately from Problem 1, we now add that an agent's position is unknown and Gaussian distributed. The agent is exposed to a risk of colliding with forbidden regions of the environment $\mathcal{F}$, such as obstacles, due to uncertainty in its position. The objective is to execute a policy that maximizes reward computed from observations, while bounding the risk of collision with the environment to be below a function of the expected reward. This chance constrained adaptive sampling problem is summarized in Problem 2.

We assume that the agent is unable to detect its position exactly, but we model its location at the time of measurement t by a multivariate Gaussian distribution with known parameters, $\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. After each measurement, an action $a_t \in \mathcal{A}$

is chosen by the policy π, which moves the agent a fixed amount $d(a_t)$, while it is subject to independent unbounded noise $\boldsymbol{w}_t \sim \mathcal{N}(0, \boldsymbol{\Sigma}_w)$. We assume actions apply identically to each location, so that $\boldsymbol{x}_{t+1} \sim \mathcal{N}(\boldsymbol{\mu}_t + d(a_t), \boldsymbol{\Sigma}_t + \boldsymbol{\Sigma}_w)$. The probability that the agent enters a forbidden region is bounded by a risk bounding function Δ applied to the expected reward.

Problem 2. *Chance Constrained Adaptive Sampling*

$$
\begin{aligned}
\pi^* \quad &= \arg\max_\pi \ \mathbb{E}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \,|\, \pi, \mathbf{o}_0\right] \\
\text{s.t.} \quad & p\left(\bigvee_{t=1}^{T} \boldsymbol{x}_t \in \mathcal{F}\right) \leq \Delta\left(\mathbb{E}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \,|\, \pi\right]\right) \\
& \mathbf{o}_{t+1} \sim p(\mathbf{o}_{t+1} \mid \mathbf{y}(\boldsymbol{x}_{t+1})) \\
& \boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \\
& \boldsymbol{x}_{t+1} \sim \mathcal{N}(\boldsymbol{\mu}_t + d(\pi(\mathbf{o}_{1:t})), \boldsymbol{\Sigma}_t + \boldsymbol{\Sigma}_w)
\end{aligned}
$$

4.4 Overview of Approach

Our approach to Problem 2 is to formulate it as a chance constrained Markov decision process (CCMDP) with a risk bounding function [113, 8], which we solve using online planning. After each observation, we solve an unconditional CCMDP using Monte Carlo tree search to ensure a policy is available at any time.

Performing this procedure naively will lead to significant conservatism, meaning that significantly less risk will be used than is permissible under the risk bounding function. Our innovations include proving that a risk bounding function is satisfied when risk and reward are averaged across different observations, construction of the unconditional online CCMDPs to reduce conservatism, and ensuring that solutions necessarily exist to the problems that are solved online.

In order to prove chance constraint satisfaction with a risk bounding function, we will leverage the Vulcan algorithm [12, 8]. The development of Vulcan proved that if every sequence of outcomes in a policy satisfies a particular local condition, then a policy satisfies a risk bounding function. This result continues to apply even if the outcomes are not finite, so we construct the constraints in our online CCMDPs to ensure that that the local condition necessarily holds for any sequence of actions

generated by online planning. This allows us to guarantee satisfaction of the risk bounding function without ever evaluating all outcomes.

We construct constraints in the online plans that match Problem 2 and limit risk as a function of reward received. In order to reduce conservatism, we measure that risk against reward computed from expected values of past observations, rather than their true values. This means that risky actions are not necessarily forbidden because of the low reward of previous actions.

Finally, in order to ensure that solutions necessarily exist to the constrained problems that we solve online, we will solve for sequences of actions that satisfy risk bounds even if no reward is received. We then show that these actions can be used to prove that a solution to the constrained online problems will always exist.

4.5 A Motivating Example

We now present a concrete, simple example to describe our approach and the difficulties associated with online chance-constrained planning. In this scenario, shown in Figure 4-1, an agent occupies some state $\boldsymbol{x}_0$. For simplicity, we do not model position as Gaussian distributed in this example, but state that certain actions simply have probability of failure.

At $\boldsymbol{x}_0$, the agent has a single action available, a_0, which moves it to the right to location $\boldsymbol{x}_1$ with no risk, where it observes $\mathbf{o}_1$, where $P[\mathbf{o}_1 = 0] = 0.2$, $P[\mathbf{o}_1 = 1] = 0.4$, and $P[\mathbf{o}_1 = 3] = 0.4$. It then has two choices available. It may move up by taking action a_1^1, which incurs no risk, and where it observes $\mathbf{o}_2^1$, with $P[\mathbf{o}_2^1 = 0] = 1$. Alternatively, it may move down by taking action a_1^2. With probability 0.1, action a_1^2 leads to failure and no observation is taken, otherwise $\mathbf{o}_2^2$ is observed. $\mathbf{o}_2^2$ is correlated with $\mathbf{o}_2^1$, so that $P[\mathbf{o}_2^2 = 0 \mid \mathbf{o}_1 = 0] = 1$, $P[\mathbf{o}_2^2 = 1 \mid \mathbf{o}_1 = 1] = 7/9$, $P[\mathbf{o}_2^2 = 6 \mid \mathbf{o}_1 = 1] = 2/9$, $P[\mathbf{o}_2^2 = 1 \mid \mathbf{o}_1 = 3] = 2/9$, $P[\mathbf{o}_2^2 = 3 \mid \mathbf{o}_1 = 3] = 7/9$.

The agent's objective is to maximize the expected sum of its measurements, so that in the language of Problem 2, $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) = \sum_{t=1}^{T} \mathbf{o}_t$. This can be constructed as a value query, modeling the observations as environment variables, letting $\zeta =$

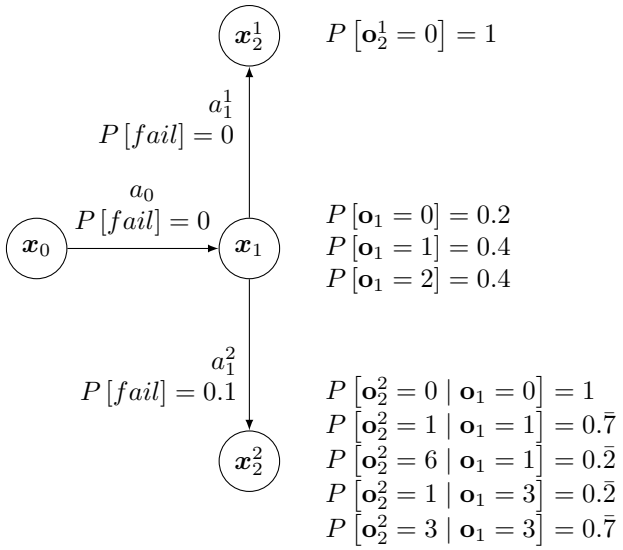


Figure 4-1: Example problem in which an agent must maximize the sum of its observations. Taking action a_2^1 leads to failure with probability 0.1.

$\sum_{t=1}^{T} \mathbf{o}_t$, and having the observations introduce no additional noise. The risk taken by the agent is constrained by a linear risk bounding function $\Delta\left(\mathbb{E}\left[\sum_{t=1}^{T} \mathbf{o}_t \mid \mathbf{o}_0\right]\right) = 0.03\ \mathbb{E}\left[\sum_{t=1}^{T} \mathbf{o}_t \mid \mathbf{o}_0\right]$.

The optimal offline policy in this example is visualized in Figure 4-2, with the rewards of different outcomes. The agent takes action a_1^1 when $\mathbf{o}_1 = 0$, and takes action a_1^2 otherwise. We can compute the expected reward achieved by this policy to be $\mathbb{E}\left[\sum_{t=1}^{T} \mathbf{o}_t \mid \mathbf{o}_0\right] = 3.28$, and the probability of failure to be 0.08. So we verify that $P\left[fail\right] = 0.08 \leq 0.0984 = 0.03 \times \mathbb{E}\left[\sum_{t=1}^{T} \mathbf{o}_t \mid \mathbf{o}_0\right]$. A policy where a_1^2 is always taken would achieve the same reward, but would have $P\left[fail\right] = 0.1$, which would violate the risk bound.

The policy in Figure 4-2 is the policy that we wish to execute by performing online planning. For example, we want the first online plan to select a_0, then we want the next online plan to select a_1^1 after $\mathbf{o}_1 = 0$ is received, and for the online plan to select a_1^2 when $\mathbf{o}_1$ is observed to be 1 or 3.

The online plans that we will construct will only have conditional outcomes for success and failure. All success states will be collected into a single state, while all failure states will be collected into a single state. For example, online CCMDPs constructed after observing $\mathbf{o}_1 = 0$ and $\mathbf{o}_1 = 1$ in the previous example would be

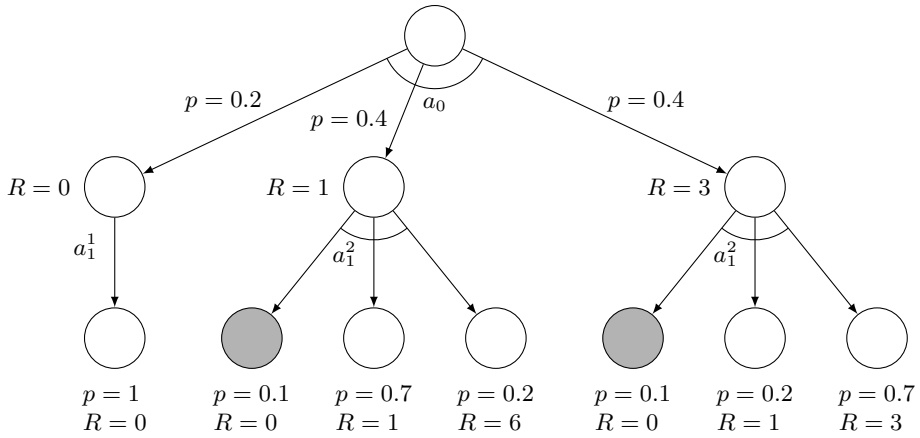
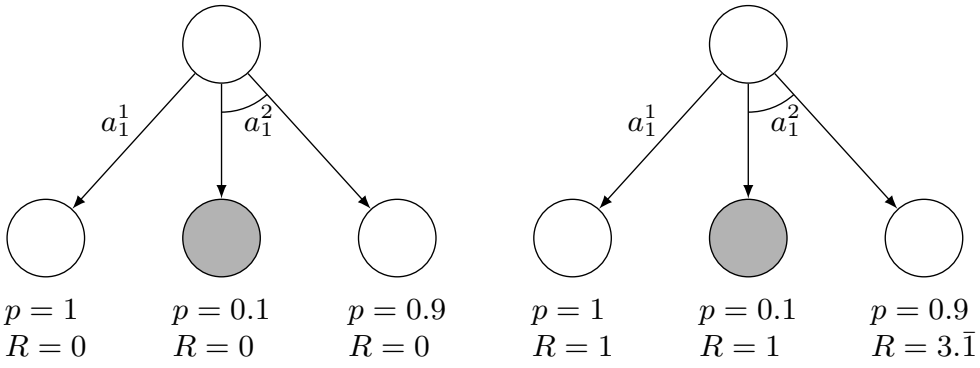


Figure 4-2: Optimal policy for the example scenario. R indicates rewards gained for actions. Dark states are failure states.



(a) Online CCMDP constructed after observing $\mathbf{o}_1 = 0$.

(b) Online CCMDP constructed after observing $\mathbf{o}_1 = 1$.

Figure 4-3: Online CCMDPs constructed in response to two different observations. Dark states are failure states. Rewards include the reward gained from previous observations at $\boldsymbol{x}_1$.

those shown in Figure 4-3. We will constrain these CCMDPs developed online so that action a_1^1 is selected in Figure 4-3a, and action a_1^2 is selected in Figure 4-3b.

One approach would be to apply the risk bounding function to rewards in the online CCMDPs. For example, we could compute that the expected reward after taking action a_1^2 in Figure 4-3a would be 0, and the risk of that action would be 0.1. After computing that $0.1 > \Delta(0)$, we would reject a_1^2 in this case, but accept a_1^1 since the probability of failure is also 0. However, using this strategy in Figure 4-3b we find that the expected reward for a_1^2 is 2.9, with risk 0.1, and $0.1 > \Delta(2.9)$, so we would also select a_1^1.

This approach is therefore conservative, because the optimal policy is not executed by selecting the optimal actions in our online CCMDPs. Likewise, we have not actually proven that this strategy will select actions that satisfy the chance constraint, shown how to generalize it when previous actions involve risk, or shown that solutions to the online CCMDPs actually exist. If, in the course of online execution, we were to construct a CCMDP without a solution, our online strategy cannot be said to actually satisfy the risk bound.

The approach detailed in this chapter answers all these issues. We use the Vulcan algorithm to prove that a modification and generalization of the approach described above actually does result in executions that respect the chance constraint, albeit still with the issue of conservatism above. We then introduce a modification to the online CCMDPs that, while still resulting in suboptimal executions in general, will execute the optimal policy in examples like this one. Then, we address the issue of solution existence, and show that we can solve a condition online which proves that all online CCMDPs can be solved if a solution exists from state s_0.

4.6 Preliminaries: Chance Constrained Markov Decision Processes with Risk Bounding Functions

Since we will model our problem as a chance-constrained Markov decision process (CCMDP) with a risk bounding function, we now review their construction and interpretation. We will use the contents of this section to formally define the problems we solve online.

4.6.1 Definition of CCMDPs with Risk Bounding Functions

A CCMDP with a risk bounding function is a method of modeling a general sequential decision process where acceptable risk is bounded as a function of reward. Here we define states, actions, and reward in CCMDPs, and specify restrictions on risk bounding functions.

A CCMDP with a risk bounding function is formally defined as the following tuple $\langle \mathcal{S}, \mathcal{C}, \mathcal{A}, \mathcal{T}, R, s_0, T, \Delta \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{C} \subseteq \mathcal{S}$ is a set of safe states, $\mathcal{A}$ is a set of actions available from each state, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ gives the probability of transitioning from a state to another by taking an action, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$ gives the reward of entering a state from another by taking an action, $s_0 \in \mathcal{C}$ is a starting state, T is the total number of actions that can be taken in the mission, and $\Delta : [0, \infty) \to [0, 1]$ is a risk bounding function which specifies the allowable probability of failure as a function of expected reward. Define a state history as an ordered sequence of states and actions,

$$h_{0:t} = \langle s_0, a_0, s_1, a_1, \ldots, s_t \rangle . \tag{4.1}$$

The objective is to find a policy $\pi^* : \mathcal{S} \to \mathcal{A}$ defined by

$$\begin{aligned} \pi^* &= \arg\max_{\pi} \ \mathbb{E}\left[u(h_{0:T}) \,|\, s_0, \pi\right] \\ \text{s.t. } & p\left(\bigvee_{t=1}^{T} S_t \notin \mathcal{C}\right) \leq \Delta\left(\mathbb{E}\left[u(h_{0:T}) \,|\, s_0, \pi\right]\right), \end{aligned} \tag{4.2}$$

where

$$u(h_{0:T}) = \sum_{t=0}^{T-1} R(s_t, a_t, s_{t+1}). \tag{4.3}$$

In our setup, states will encode information about agent position and previous observations, and safe states will be those where the position is outside the limits of obstacles. Reward will be determined by a query objective, and the risk bounding function will constrain how much risk of collision with the environment is acceptable as a function of the expected query objective over an executed policy.

4.6.2 Interpretation of Risk Bounding Functions

An unconstrained MDP considers failure in the sense that the expected reward decreases as failure probability increases. However, the optimal unconstrained policy may result in a probability of failure that is arbitrarily close to 1, while simultaneously

placing no constraint on how large the reward must be in this case. This behavior is undesirable for a mission designer, for whom the expense of collecting and repairing the vehicle may not be worth the data obtained.

We encode the tolerance for risk of a mission designer through the risk bounding function, which specifies the allowable probability of failure of a policy as a function of the expected reward. Our solution method will require that Δ is a non-decreasing concave function so that we can make use of certain bounds on probability of failure. These are not particularly restrictive assumptions; they encode that we will not forbid a mission from using the same amount of risk as another that achieves less reward, and that progressively additional reward is worth less additional risk, both of which are reasonable assumptions for risk tolerance.

Risk bounding functions generalize the notion of a single risk bound that appears in the chance constrained planning literature [99]. The idea of a functional representation of failure tolerance is particularly important in exploration, where the increase in reward with risk is unknown. This idea is illustrated in Figure 4-4, where in (a) a small increase in risk allows the agent to move closer to the obstacle and leads to a large increase in reward, while in (b) it has no effect. A mission designer may prefer a slightly higher risk bound in (a), but a lower risk bound in (b). A risk bounding function encodes this tolerance without the need to know the relationship between risk and reward in the environment, only the price in failure probability that the mission designer is willing to pay for reward.

There does exist a relationship between risk and reward chance constrained motion planning problems, such as navigating autonomous underwater vehicles or autonomous cars with position uncertainty to specific locations. In those problems, increases in risk bounds lead to reductions in transit time or required control, as vehicles are able to travel closer to obstacles. It is natural to set the level of permissible risk based on how damaging loss of the vehicle would be. So long as the risk bound is not too low, the final waypoint will always be reached, and changing the risk bound controls how quickly or efficiently the goal is achieved. Adaptive sampling missions are different because there is not a single goal that will or will not be achieved. The

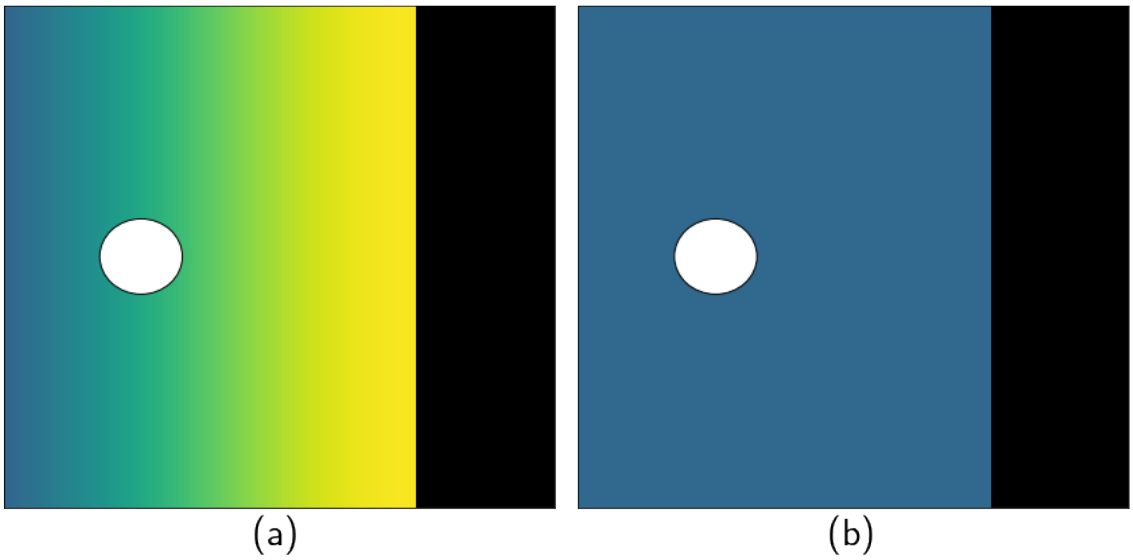


Figure 4-4: An agent (white) close to an obstacle (black) with (a) high reward close to the obstacle and (b) uniform reward. Reward of observation is indicated by the color of the environment, with brighter colors worth more.

success of a mission depends on the amount of information it gathers, and so success is directly determined by the reward, which depends on permissible risk. For this reason, we wish to make the relationship between risk and reward explicit in adaptive sampling problems.

It is important to note that the chance constraint uses expectations of failure and reward over the entire policy. Dangerous actions with low reward are permitted if high reward is achieved on other measurement histories. This is important, because an unlikely realization of the environment may result in minimum rewards for all actions, but it may not be possible to execute actions for which the probability of failure is below $\Delta(0)$.

4.7 Preliminaries: The Vulcan Algorithm

In our approach, we will use the previously developed Vulcan algorithm [8], which we now describe here. In particular, we will use a theorem from Ayton and Williams [8] which states that a risk bounding function is satisfied for a policy if every a quantity called the sequence execution risk is bounded for every possible outcome of

the possible. We will then use this result to prove that the plans produce and execute online satisfy the risk bounding function by proving that sequence execution risk is bounded for every possible outcome of the online plans.

Vulcan is based on the *Upper Confidence Bound applied to Trees* (UCT) algorithm for MDPs, which uses previous samples of the search tree to guide future samples towards promising solutions [72]. In UCT, random sampling is performed to build a search tree. Rollouts are performed from a root state, with each rollout selecting a single action from a state, choosing a single outcome of the chosen action, and repeating this process until a leaf state is reached. The action selection rule used by UCT is that on the first $|\mathcal{A}|$ times a state is sampled, actions that have never been selected are chosen. On subsequent samples, the action is chosen according to

$$a_t = \arg\max_a \; Q(s_t, a) + \sqrt{\frac{2 \log N_{s_t}}{N_{s_t,a}}}. \tag{4.4}$$

$Q(s_t, a_t)$ is an empirical average of the reward achieved on rollouts that took action a_t from s_t, N_{s_t} is the number of samples taken at state s_t, and N_{s_t,a_t} is the number of samples of a_t from s_t.

In order to apply UCT to CCMDPs, the challenge is that MCTS only considers rollouts individually, and does not explicitly consider expectations of reward over multiple candidate policies. This means that if the highest reward policy does not satisfy the risk bound, UCT does not provide a means to estimate the reward of the next best policy.

To circumvent this issue, Vulcan introduces a sufficient condition for satisfaction of the risk bounding function that applies to each sequence of states individually. In this way, satisfaction of the risk bound can be determined for each sequence of actions on the rollouts that sample them, rather than computing expected reward for multiple policies. Vulcan defines the sequence execution risk *ser* of a state history

$h_{0:t} = \langle s_0, a_0, s_1, a_1, \ldots, s_t \rangle$ as

$$ser(h_{0:T}) = \begin{cases} \frac{p\left(\bigvee_{t=1}^{T} S_t \notin \mathcal{C} \,\middle|\, s_0, a_{0:T-1}\right)}{1-p\left(\bigvee_{t=1}^{T} S_t \notin \mathcal{C} \,\middle|\, s_0, a_{0:T-1}\right)} & \text{no failures} \\ \\ 0 & \text{otherwise} \end{cases}. \tag{4.5}$$

In this equation, $p\left(\bigvee_{t=1}^{T} S_t \notin \mathcal{C} \,\middle|\, s_0, a_{0:T-1}\right)$ is the combined probability that a failure state is entered by executing a_0 from s_0, executing a_1 from s_1, up to executing a_{T-1} from s_{T-1}. This can be thought of as a measure of risk for the individual state history. *ser* is defined to be 0 for state histories that encounter failures, which is a technical choice made so that actions that lead to failures early in the policy are not found to violate the risk bound. The denominator of *ser* is then a normalization, to account for the outcomes in which *ser* is 0.

UCT proceeds as in the work of Kocsis and Szepesvári [72], by performing rollouts according to the previous action selection rule, and computing $Q(s_t, a_t)$. Where Vulcan differs is that upon reaching a safe state at the planning horizon T, it adds an additional condition that the entire state history in the rollout is checked to satisfy

$$ser(h_{0:T}) \leq \Delta\left(v(h_{0:T})\right), \tag{4.6}$$

where v is any function that satisfies $\mathbb{E}\left[v(h_{0:T}) \,|\, s_0, \pi\right] \leq \mathbb{E}\left[u(h_{0:T}) \,|\, s_0, \pi\right]$. By the definition of *ser* that is defined to be 0 for any histories with a failure, the equation (4.6) is satisfied for any histories ending in failure states, which is desirable so that risks with low immediate rewards are allowed if they lead to high rewards later. If (4.6) is not satisfied or no actions remain from a state, then the last action is deleted from the search tree and a new sample is taken from s_0.

Theorem 1 of Ayton and Williams [8] shows that any policy found under this strategy is guaranteed to satisfy the chance constraint. The proof follows from the fact that the expectation of *ser* across all state histories in a policy equals the total

probability of failure, so that

$$
\begin{aligned}
p\left(\bigvee_{t=1}^{T} S_t \notin \mathcal{C} \,\middle|\, s_0, \pi\right) &= \mathbb{E}\left[ser\left(h_{0:T}\right) \mid s_0, \pi\right] \\
&\leq \mathbb{E}\left[\Delta\left(v(h_{0:T})\right) \mid s_0, \pi\right] \\
&\leq \Delta\left(\mathbb{E}\left[v(h_{0:T}) \mid s_0, \pi\right]\right) \\
&\leq \Delta\left(\mathbb{E}\left[u(h_{0:T}) \mid s_0, \pi\right]\right)
\end{aligned} \tag{4.7}
$$

by Jensen's inequality for non-decreasing concave Δ. The resultant policy is suboptimal, but the advantage in this context is that (4.6) can be applied to each state history without knowledge of the others.

In order to use Vulcan in an online planning framework, we will use the fact that the executed actions are guaranteed to satisfy the chance constraint if a state history satisfying (4.6) *can be* found after any set of observations, *even if they have not been computed explicitly.* Therefore, our approach will be to construct the online planning problems so that the actions selected by solving those problems will necessarily result in state histories that will satisfy (4.6).

4.8 Modeling Chanced Constrained Adaptive Sampling as a CCMDP

In this section, we describe how we model Problem 2 as a chance-constrained Markov decision process with a risk bounding function. We detail how we capture agent state, how we define state transitions, and how we define reward.

The CCMDP constructed here produces a unique state for each observation, so that an optimal policy can respond to differences in observations. This CCMDP is introduced so that we can analyze the behavior of the actions that we will executed online, and show that they satisfy the risk bounding function when averaged over all outcomes. However, when we solve this problem online, we will construct smaller CCMDPs with a single state that captures all possible observations at a location.

4.8.1 State Formulation

Since the agent is never aware of or capable of responding to its true position history, a state s_t includes a full history of position distributions. These are characterized by vectors of mean positions $\boldsymbol{\mu}_{0:t}$ and covariances $\boldsymbol{\Sigma}_{0:t}$. In addition, the agent responds to its entire history of observations $\mathbf{o}_{0:t}$, and whether a failure has occurred, which is indicated with a binary variable F_t that is zero when no failure has occurred and one otherwise, so a state is defined as

$$s_t = \langle \boldsymbol{\mu}_{0:t}, \boldsymbol{\Sigma}_{0:t}, \mathbf{o}_{0:t}, F_t \rangle . \tag{4.8}$$

The initial state defines the initial mean, covariance, and prior measurements, and is assumed to be safe. From the perspective of a CCMDP, $\mathcal{C}$ is the set of all states where $F_t = 0$, and failure states satisfy $F_t = 1$. We consider only a single failure state resulting from each state and action, which encompasses all outcomes in which $x_{t+1} \in \mathcal{F}$.

4.8.2 State Transitions

It is difficult to compute the true probability of collision of an agent with a Gaussian state distribution with an arbitrary environment. $\boldsymbol{x}_t$ is described by a Gaussian distribution, but when conditioned on the safety of previous states it is not Gaussian in general, since this suggests that $\boldsymbol{x}_k \notin \mathcal{F}$ for all $k \leq t$. In our CCMDPs, we will instead conservatively overestimate the value.

A state s_t encodes distributions over $\boldsymbol{x}_{0:t}$, rather than exact values of position. For appropriate mean and covariances, and from a safe state s_t to another safe state

s_{t+1}, probability of transition can be computed as

$$\begin{aligned}
\mathcal{T}(s_t, a_t, s_{t+1}) &= p(\mathbf{o}_{t+1}, F_{t+1} = 0 \mid s_t, a_t) \\
&= p(F_{t+1} = 0 \mid s_t, a_t)\ p(\mathbf{o}_{t+1} \mid F_{t+1} = 0, s_t, a_t) \\
&= p\left(F_{t+1} = 0 \mid s_t, a_t\right) \\
&\quad \times \int p(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t}, \boldsymbol{x}_{0:t+1}, \mathbf{F}_{0:t+1} = \mathbf{0})\ p(\boldsymbol{x}_{0:t+1} \mid \mathbf{F}_{0:t+1} = \mathbf{0})\ d\boldsymbol{x}_{0:t+1} \\
&= p\left(F_{t+1} = 0 \mid s_t, a_t\right)\ \int p(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t}, \boldsymbol{x}_{0:t+1})\ p(\boldsymbol{x}_{0:t+1} \mid \mathbf{F}_{0:t+1} = \mathbf{0})\ d\boldsymbol{x}_{0:t+1}.
\end{aligned} \tag{4.9}$$

where the last line follows from the facts that s_t, and a_t encode that $\mathbf{F}_{0:t+1} = \mathbf{0}$, that the exact values of $\boldsymbol{x}_{0:t+1}$ are marginalized out from states, and that

$$p(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t}, \boldsymbol{x}_{0:t+1}, \mathbf{F}_{0:t+1} = \mathbf{0}) = p(\mathbf{o}_{t+1} \mid \mathbf{o}_{0:t}, \boldsymbol{x}_{0:t+1}). \tag{4.10}$$

The probability from a safe state s_t to failure state s_{t+1} is

$$\mathcal{T}(s_t, a_t, s_{t+1}) = p(F_{t+1} = 1 \mid s_t, a_t). \tag{4.11}$$

It is useful to think of failure states as being terminal, with no actions available from them. Formally, we define a single action as available which always leads back to the same state, netting zero reward.

4.8.3 Reward Function

We model the agent as receiving reward immediately after measurement, when possible. Upon collision with the environment, which is described by the forbidden region $\mathcal{F}$, it is unable to perform the measurement, but previous rewards are not lost. For an underwater vehicle, this case occurs when collision triggers a mission abort and surface sequence, which takes the vehicle out the field until a diagnostic can be run and parts can be replaced, but previous measurements are recoverable. Successful abort sequences after a collision are common for slow vehicles with line of sight to the surface, and we leave the less common case where data is lost in collision to future

research.

In practice, this means that when $\boldsymbol{x}_{t+1}$ is in the forbidden region, the reward received is $g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$. As a result, for failure states s_{t+1}, we have $R(s_t, a_t, s_{t+1}) = g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$. Safe states s_{t+1} receive zero reward unless they are leaf states, in which case we have $R(s_{T-1}, a_{T-1}, s_T) = g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$. As a result, for a sequence of safe states, $u(h_{0:T}) = g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$.

4.9 Risk Approximations

In this section, we detail how we compute bounds on risk of collision with the environment, and then use those bounds to develop bounds for sequence execution risk. Most of the following development of bounds on risk were previously presented by Ono and Williams [99], but our use of these expressions to develop bounds on *ser* are novel.

Since probability of failure is difficult to compute exactly, we follow Ono and Williams [99] to develop a conservative bound using Boole's inequality.

$$\begin{aligned} p\left(\bigvee_{t=1}^{T} S_t \notin \mathcal{C} \,\middle|\, s_0, a_{0:T-1}\right) &= p\left(\bigvee_{t=1}^{T} \boldsymbol{x}_t \in \mathcal{F} \,\middle|\, s_0, a_{0:T-1}\right) \\ &\leq \sum_{t=1}^{T} p\left(\boldsymbol{x}_t \in \mathcal{F} \mid \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\right), \end{aligned} \tag{4.12}$$

where

$$p(\boldsymbol{x}_t \in \mathcal{F} \mid \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) = \int_{\mathcal{F}} \mathcal{N}\left(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\right) dx_t. \tag{4.13}$$

(4.13) remains computationally intensive for arbitrary obstacles, and sampling based approximations can underestimate the risk. Instead, we use an estimation that is guaranteed to be conservative by enclosing $\mathcal{F}$ with a union of N_F convex polytopes,

$$\mathcal{F} \subseteq \bigcup_{i=1}^{N_F} \mathcal{F}_i, \tag{4.14}$$

where polytope $\mathcal{F}_i$ may be described as an intersection of half-spaces based on each

of its E_i edges,

$$\mathcal{F}_i = \left\{ \boldsymbol{x} \;\middle|\; \bigwedge_{j=1}^{E_i} \boldsymbol{h}_{ij}^T \boldsymbol{x} \geq g_{ij} \right\}, \tag{4.15}$$

for vector $\boldsymbol{h}_{ij}$ and scalar g_{ij}.

Using Boole's inequality again, we have

$$p(\boldsymbol{x}_t \in \mathcal{F} \mid \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \leq \sum_{i=1}^{N_F} p(\boldsymbol{x}_t \in \mathcal{F}_i \mid \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t). \tag{4.16}$$

Let $\mathcal{E}_i(\boldsymbol{\mu}_t)$ be the set of half-space indices of $\mathcal{F}_i$ that the mean state $\boldsymbol{\mu}_t$ lies outside,

$$\mathcal{E}_i(\boldsymbol{\mu}_t) := \left\{ j \;\middle|\; \boldsymbol{h}_{ij}^T \boldsymbol{\mu}_t < g_{ij} \right\}. \tag{4.17}$$

Then we bound the probability of collision by the minimum probability of entering one of the half-spaces in $\mathcal{E}_i(\boldsymbol{\mu}_t)$:

$$p(\boldsymbol{x}_t \in \mathcal{F}_i \mid \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \leq \min_{j \in \mathcal{E}_i(\boldsymbol{\mu}_t)} p(\boldsymbol{h}_{ij}^T \boldsymbol{x}_t \geq g_{ij}), \tag{4.18}$$

$$p(\boldsymbol{h}_{ij}^T \boldsymbol{x}_t \geq g_{ij}) = \frac{1}{2} + \frac{1}{2} \text{erf}\left(\frac{\boldsymbol{h}_{ij}^T \boldsymbol{\mu}_t - g_{ij}}{\sqrt{2 \boldsymbol{h}_{ij}^T \boldsymbol{\Sigma}_t \boldsymbol{h}_{ij}}} \right). \tag{4.19}$$

The intuition for (4.18) is given in Figure 4-5, while (4.19) follows from a Gaussian cumulative density function. A convex decomposition of the forbidden regions can be computed prior to the start of planning.

In order to use these bounds on probability of failure within sequence execution risk, we define p_f as the sum of bounds on probabilities of collision with each obstacle in the environment,

$$p_f(s_t) = \sum_{i=1}^{N_F} \min_{j \in \mathcal{E}_i(\boldsymbol{\mu}_t)} p(\boldsymbol{h}_{ij}^T \boldsymbol{x}_t \geq g_{ij}). \tag{4.20}$$

Then it follows from (4.5), (4.12), and (4.16) that

$$ser(h_{0:T}) \leq \frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)}. \tag{4.21}$$

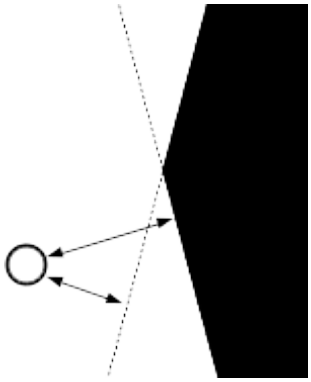


Figure 4-5: Probability of collision with a convex obstacle is less than the minimum probability of crossing a line defined by its edges.

The approach we take is then to sample over potential trajectories, and verify that each potential trajectory satisfies

$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} \leq \Delta\left(v(h_{0:T})\right) \tag{4.22}$$

for some choice of function v.

4.10 Solution Procedure

Even using Vulcan, Problem 2 is too large to find an explicit solution, as Vulcan requires every outcome of the policy to be sampled in order to guarantee the chance constraint is satisfied. Instead, we will construct plans online that only condition on success or failure, in which each state corresponds to a choice of actions, and will include all possible observations.

After receiving an observation, we replan from state s_t, constructing a CCMDP starting at s_t that solves for a plan conditioned on the observations seen so far. In the online CCMDP, action a_k results in failure with probability $p_f(s_{k+1})$, or a single success state, as opposed to distinct states for each observation as in Section 4.8.1. The reward for entering the safe state is bounded by the expected reward from all observations resulting from the sequence of actions found. The first action of the policy is executed, then replanning is performed from the true s_{t+1}. We only ever

produce unconditional plans, because the mission ends immediately after any failure that occurs, and only one success state follows each action. However, the plans are constructed to depend on all observations so far, and so the actions selected are adaptive to the observations that have been received. We can then determine whether the risk bounding function is satisfied for the expected risk and expected reward over all possible executions.

Online planning is a standard technique. Planning this way allows significantly smaller problems to be solved, with the disadvantages that the executed actions are suboptimal, and that computational effort must be expended online. The innovation in our approach is that we construct our online plans to guarantee that Vulcan's condition (4.6) *must* be satisfied for every outcome that occurs, so that the risk bound is necessarily satisfied. This involves carefully selecting the bounds we impose on the online CCMDPs, and guaranteeing that a solution will always exist for the online CCMDP that we construct. We show in this section that this can be done.

4.10.1 Construction of Online CCMDPs

We will construct the online CCMDPs so that outcomes of an action that do not result in failure are contained in the same state. This means that all actions will have at most two outcomes in the online CCMDPs; a failure state where execution immediately ends, and a success state where the action is taken and an observation is received. The reward for entering success states will be based on the expected reward across the different observations that could be received.

Consider planning online from a state s_t, after having received past observations $\mathbf{o}_{0:t}$. Like in Chapter 2, a sequence of safe states up to the plan horizon T will receive reward $\mathbb{E}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right]$ for value and probability objectives, while information objectives receive $\mathbb{E}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right] + g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$.

For sequences of states ending in failure states $s_{t'}$, meanwhile, value and probability objectives receive reward $\mathbb{E}\left[g(\mathbf{o}_{t+1:t'} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right]$, while information objectives receive $\mathbb{E}\left[g(\mathbf{o}_{t+1:t'} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right] + g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)$. Since all other states receive no reward, these results are also the values of $u(h_{0:T})$ and $u(h_{0:t'+1})$.

The observations $\mathbf{o}_t$ are drawn from the distribution at the location $\boldsymbol{\mu}_t$. This approximation produces a small amount of error under the reasonable assumption that position distribution length scales are small compared to length scales of spatial correlations in the model.

The reward functions described here do not differ substantially from the previous chapter, but we have described them to contrast them against the constraints we will impose in the online CCMDPs that are described in this section.

4.10.2 Basic Chance Constraint Satisfaction

In this section, we show how placing constraints on the online CCMDPs ensures that the chance constraint is satisfied over all possible executions. We wish to ensure that the chance constraint is satisfied, meaning that the expected risk across all plan executions is bounded as a function of the expected reward across all executions. When we plan online from a state s_t, the agent has already selected actions a_0 through a_{t-1} and observed the outcomes of those actions by receiving observations $\mathbf{o}_{1:t}$. In doing so, the agent has moved through a sequence of states that were modeled by the fully conditional CCMDP in Section 4.8. But by performing online planning, we then are not generating the states $s_{t+1:T}$ in the conditional CCMDP, but instead using a CCMDP with a smaller state space.

The key to enforcing the chance constraint is realizing that placing a bound of the form (4.22) on the online CCMDP and solving for an unconditional plan $a_{t:T-1}$ will also place a bound on the sequence execution risk of all states that could be reached in the conditional CCMDP by following $a_{t:T-1}$ from s_t. After t actions, an online CCMDP is constructed for each s_t that could be reached, and we can view the actions selected online up to s_t, followed by the actions found by online CCMDPs at state s_t as a policy in the conditional CCMDP. We can analyze the properties of that policy, and construct the online CCMDPs so that the chance constraint is necessarily satisfied.

To place bounds on the online CCMDPs, we can select an appropriate function v, so that (4.6) holds for all such $\mathbf{o}_{t+1:T}$ along with the $\mathbf{o}_{1:t}$ that were already observed. By

constructing each online CCMDP to satisfy this condition, we guarantee the chance constraint without needing to solve the conditional CCMDP.

Let us now demonstrate this more concretely, and consider choices of v that we could use. An intuitive choice when planning in the online CCMDP from state s_t would be to bound the sequence execution risk using the reward that is computed for states in the online CCMDP. For each leaf state, this would mean that when the objective is a value or probability objective, the state histories in the online CCMDP satisfy

$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} \leq \Delta\left(\mathbb{E}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right]\right), \tag{4.23}$$

while for information objectives they satisfy

$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} \leq \Delta\left(\mathbb{E}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right] + g(\mathbf{o}_{1:t} \mid \mathbf{o}_0)\right). \tag{4.24}$$

If these conditions are not satisfied, we delete the last state of a state sequence from the search tree, so that search is only performed over those states where the conditions are satisfied. Among those plans where these conditions are satisfied, we select the one with the highest expected reward.

Now let us consider a state history in the fully conditional CCMDP that was reached by following $a_{t:T-1}$ from s_t, and received some observations $\mathbf{o}_{1:t}$. The sequence execution risk of that state history is still bounded by $\sum_{t=1}^{T} p_f(s_t) / \left(1 - \sum_{t=1}^{T} p_f(s_t)\right)$, as computed in the online CCMDP, because the probability of entering a failure state is the same in the fully conditional CCMDP and online CCMDP. For that sequence, we need to demonstrate that

$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} \leq \Delta\left(v(h_{0:T})\right) \tag{4.25}$$

for some $v(h_{0:T})$ such that $\mathbb{E}\left[v(h_{0:T}) \,|\, s_0, \pi\right] \leq \mathbb{E}\left[u(h_{0:T}) \,|\, s_0, \pi\right]$ where $u(h_{0:T}) = g(\mathbf{o}_{1:T} \mid \mathbf{o}_0)$. But this necessarily holds, because we showed in Chapter 2 that

$$\mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[\mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right] \mid \mathbf{o}_0\right] = \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right] \tag{4.26}$$

for value and probability objectives, while

$$\mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[\mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right] + g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right] = \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right] \quad (4.27)$$

for information objectives. Therefore the condition we require is satisfied, because $\mathbb{E}\left[v(h_{0:T}) \,|\, s_0, \pi\right] = \mathbb{E}\left[u(h_{0:T}) \,|\, s_0, \pi\right]$ when using

$$\begin{aligned} v(h_{0:T}) &= \mathbb{E}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right], && \text{value or probability objectives} \\ v(h_{0:T}) &= \mathbb{E}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right] + g(\mathbf{o}_{1:t} \mid \mathbf{o}_0), && \text{information objectives.} \end{aligned} \quad (4.28)$$

Example

Let us make this explicit using our previous example, where we had a value objective for which $g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) = \sum_{t=1}^{T} \mathbf{o}_t$. Execution satisfies the chance constraint if, for each safe state history in the conditional CCMDP, we can show that

$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} \leq \Delta\left(v(h_{0:T})\right),$$

where $\mathbb{E}\left[v(h_{0:T}) \,|\, s_0, \pi\right] \leq \mathbb{E}\left[\sum_{t=1}^{T} \mathbf{o}_t \,\middle|\, \mathbf{o}_0, \pi\right]$. In the online CCMDP, we produce a plan from state s_t where every safe state history,

$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} \leq \Delta\left(\mathbb{E}\left[\sum_{k=1}^{T} \mathbf{o}_k \,\middle|\, \mathbf{o}_{0:t}\right]\right).$$

If this constraint holds for the online plan that is generated in response to each $\mathbf{o}_{1:t}$, then we have that for each state history in the conditional CCMDP, that

$$ser(h_{0:T}) \leq \frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} \leq \Delta\left(\mathbb{E}\left[\sum_{k=1}^{T} \mathbf{o}_k \,\middle|\, \mathbf{o}_{0:t}\right]\right).$$

Since $\mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[\mathbb{E}_{\mathbf{o}_{t+1:T}|\mathbf{o}_{0:t}}\left[\sum_{k=1}^{T}\mathbf{o}_k \mid \mathbf{o}_{0:t}\right] \mid \mathbf{o}_0\right] = \mathbb{E}_{\mathbf{o}_{1:T}|\mathbf{o}_0}\left[\sum_{k=1}^{T}\mathbf{o}_k \mid \mathbf{o}_0\right]$, then we have that $\mathbb{E}\left[v(h_{0:T}) \mid s_0, \pi\right] \leq \mathbb{E}\left[u(h_{0:T}) \mid s_0, \pi\right]$, and Theorem 1 of Ayton and Williams [8] guarantees the chance constraint is satisfied. We did *not* verify that $ser(h_{0:T}) \leq \Delta\left(u(h_{0:T})\right) = \Delta\left(\sum_{t=1}^{T}\mathbf{o}_t\right)$ for every state history in the conditional CCMDP, and this turns out to be unnecessary.

For example, let us say that the agent observed $\mathbf{o}_1 = 0$ in the motivating example, and we plan online from state s_1, as in Figure 4-3a. For the safe state history $\langle s_0, a_0, s_1, a_1^1, s_2\rangle$, we have

$$\begin{aligned}
\sum_{t=1}^{T} p_f(s_t) &= 0 \\
\mathbb{E}\left[\mathbf{o}_1 + \mathbf{o}_2 \mid \mathbf{o}_1\right] &= 0 \\
\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} = 0 \leq 0 &= \Delta\left(\mathbb{E}\left[\sum_{k=1}^{T}\mathbf{o}_k \;\middle|\; \mathbf{o}_{0:t}\right]\right).
\end{aligned}$$

Therefore, the conditions above hold. For the safe state history $\langle s_0, a_0, s_1, a_1^2, s_2\rangle$, we have

$$\begin{aligned}
\sum_{t=1}^{T} p_f(s_t) &= 0.1 \\
\mathbb{E}\left[\mathbf{o}_1 + \mathbf{o}_2 \mid \mathbf{o}_1\right] &= 0 \\
\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} = 0.\bar{1} > 0 &= \Delta\left(\mathbb{E}\left[\sum_{k=1}^{T}\mathbf{o}_k \;\middle|\; \mathbf{o}_{0:t}\right]\right),
\end{aligned}$$

so the conditions above are violated. As a result, action a_1^1 is selected after receiving $\mathbf{o}_1 = 0$.

Now let's consider that the agent observed $\mathbf{o}_1 = 1$. For the safe state history

$\langle s_0, a_0, s_1, a_1^1, s_2 \rangle$,

$$\sum_{t=1}^{T} p_f(s_t) = 0$$
$$\mathbb{E}\left[\mathbf{o}_1 + \mathbf{o}_2 \mid \mathbf{o}_1\right] = 1$$
$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} = 0 \leq 0.03 = \Delta\left(\mathbb{E}\left[\sum_{k=1}^{T} \mathbf{o}_k \;\middle|\; \mathbf{o}_{0:t}\right]\right).$$

so the conditions above hold. For the safe state history $\langle s_0, a_0, s_1, a_1^2, s_2 \rangle$, we have

$$\sum_{t=1}^{T} p_f(s_t) = 0.1$$
$$\mathbb{E}\left[\mathbf{o}_1 + \mathbf{o}_2 \mid \mathbf{o}_1\right] = 3.\bar{1}$$
$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} = 0.\bar{1} > 0.09\bar{3} = \Delta\left(\mathbb{E}\left[\sum_{k=1}^{T} \mathbf{o}_k \;\middle|\; \mathbf{o}_{0:t}\right]\right).$$

so the conditions above is still violated. As a result, action a_1^1 is selected.

Finally, consider that the agent observed $\mathbf{o}_1 = 3$. For the safe state history $\langle s_0, a_0, s_1, a_1^2, s_2 \rangle$,

$$\sum_{t=1}^{T} p_f(s_t) = 0.1$$
$$\mathbb{E}\left[\mathbf{o}_1 + \mathbf{o}_2 \mid \mathbf{o}_1\right] = 5.\bar{5}$$
$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} = 0.\bar{1} < 0.1\bar{6} = \Delta\left(\mathbb{E}\left[\sum_{k=1}^{T} \mathbf{o}_k \;\middle|\; \mathbf{o}_{0:t}\right]\right).$$

The conditions above are satisfied, and a_1^2 results in higher reward than a_1^1, so a_1^2 will be selected.

Overall, performing online planning in this way will execute a_1^1 after observing $\mathbf{o}_1 = 0$ or 1, and a_1^2 after observing $\mathbf{o}_1 = 3$. The expected reward of this execution is 2.52, and the risk taken is 0.004, which is less than $\Delta(2.52) = 0.0756$, so the chance constraint is satisfied overall.

4.10.3 Less Conservative Chance Constraint Satisfaction

In the previous section we showed that the solutions to the online CCMDPs that we construct necessarily result in execution that satisfies the chance constraint. However, we also showed through the example above that actions executed can be suboptimal. Here, we show how to find a condition that results in behavior that is less suboptimal for certain problems.

The reason for suboptimality in the example above, is that in the optimal policy, observations that follow $\mathbf{o}_1 = 2$ generate significant reward. That reward is enough to justify the additional risk taken when $\mathbf{o}_1 = 1$, even though the rewards received from taking action a_1^2 after observing $\mathbf{o}_1$ are not enough to justify the reward alone. The choice of v made above may not satisfy (4.22) when observations are low, even if there is enough reward on other branches to justify the risk.

Instead, for value or probability objectives for which $\mathbb{E}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right]$ can be written in the form

$$\mathbb{E}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right] = \sum_{k=1}^{T} \mathbb{E}\left[\tilde{g}_k(\mathbf{o}_{0:k}) \mid \mathbf{o}_{0:k}\right] \tag{4.29}$$

for some functions $\tilde{g}_k(\mathbf{o}_{0:k})$, we are able to share reward between different branches of online CCMDP when determining if risk is satisfied. An examples is a value objective with $g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) = \sum_{k=1}^{T} \mathbf{o}_k$, for which $\tilde{g}_k(\mathbf{o}_{0:k}) = \mathbf{o}_k$.

When solving the online CCMDPs from s_t, we require that safe state histories satisfy

$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} \leq \Delta \left(\sum_{k=t+1}^{T} \mathbb{E}\left[\tilde{g}_k(\mathbf{o}_{0:k}) \mid \mathbf{o}_{0:t}\right] + \sum_{k=1}^{t} \mathbb{E}\left[\tilde{g}_k(\mathbf{o}_{0:k}) \mid \mathbf{o}_{0:k-1}\right] \right). \tag{4.30}$$

The intuition behind (4.30) is to use expected reward computed with fewer observations than have actually been taken. This way, when low reward is achieved but it was possible to achieve greater reward on unobserved outcomes, the reward passed into the risk bounding function will be increased, allowing greater risk. Note that in (4.30), each expectation on the right depends on $\mathbf{o}_{0:k-1}$ rather than the observations

$\mathbf{o}_{0:k}$. In this way, checking whether a state history satisfies the conditions necessary to satisfy the risk bound depends upon an observation $\mathbf{o}_k$ not directly in the state history.

This corresponds to the following choice of v

$$v(h_{0:t}) = \sum_{k=t+1}^{T} \mathbb{E}\left[\tilde{g}_k(\mathbf{o}_{0:k}) \mid \mathbf{o}_{0:t}\right] + \sum_{k=1}^{t} \mathbb{E}\left[\tilde{g}_k(\mathbf{o}_{0:k}) \mid \mathbf{o}_{0:k-1}\right] \tag{4.31}$$

To show that the risk bound is satisfied, we must show that $\mathbb{E}\left[v(h_{0:T}) \,|\, s_0, \pi\right] \leq \mathbb{E}\left[u(h_{0:T}) \,|\, s_0, \pi\right]$. Here, the expectation is taken with respect to all observations that would be received by executing the first actions of online plans found up to time step t, then executing the actions $a_{t:T-1}$ found by the online CCMDPs at state s_t. The condition on expectation holds because the expectation of $v(h_{0:T})$ conditioned on $\mathbf{o}_0$ is the same as the expectation of $u(h_{0:T})$ conditioned on $\mathbf{o}_0$. More precisely,

$$\begin{aligned}
\mathbb{E}\left[v(h_{0:T}) \mid s_0, \pi\right] &= \mathbb{E}\left[\sum_{k=t+1}^{T} \mathbb{E}\left[\tilde{g}_k(\mathbf{o}_{0:k}) \mid \mathbf{o}_{0:t}\right] + \sum_{k=1}^{t} \mathbb{E}\left[\tilde{g}_k(\mathbf{o}_{0:k}) \mid \mathbf{o}_{0:k-1}\right] \,\middle|\, \mathbf{o}_0\right] \\
&= \mathbb{E}\left[\sum_{k=t+1}^{T} \mathbb{E}\left[\tilde{g}_k(\mathbf{o}_{0:k}) \mid \mathbf{o}_{0:t}\right] + \sum_{k=1}^{t} \mathbb{E}\left[\tilde{g}_k(\mathbf{o}_{0:k}) \mid \mathbf{o}_{0:k}\right] \,\middle|\, \mathbf{o}_0\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right] \mid \mathbf{o}_0\right] \\
&= \mathbb{E}\left[g(\mathbf{o}_{1:T} \mid \mathbf{o}_0) \mid \mathbf{o}_0\right] \\
&= \mathbb{E}\left[u(h_{0:T}) \,|\, s_0, \pi\right]
\end{aligned} \tag{4.32}$$

and the chance constraint is satisfied. The second line follows from the fact that the outer expectation is over all observations that could be received following π in the conditional CCMDP. The expectation over the policy of an expectation conditioned on $\mathbf{o}_{0:k}$ is the same as the expectation over the policy of an expectation conditioned on $\mathbf{o}_{0:k-1}$.

Unlike (4.23), it is *not* necessarily true that sequence execution risk is bounded by the risk bounding function applied to the observations that occur on the state history in (4.30), so that reward can be shared between different branches.

Example

To make this concrete, consider the example of maximizing the observations received. The form of (4.30) is then

$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} \leq \Delta \left(\sum_{k=t+1}^{T} \mathbb{E}\left[\mathbf{o}_k \mid \mathbf{o}_{0:t}\right] + \sum_{k=1}^{t} \mathbb{E}\left[\mathbf{o}_k \mid \mathbf{o}_{0:k-1}\right] \right).$$

Let us say that the agent observed $\mathbf{o}_1 = 0$ in the motivating example. For the safe state history $\langle s_0, a_0, s_1, a_1^1, s_2 \rangle$, we have

$$\begin{aligned}
\sum_{t=1}^{T} p_f(s_t) &= 0 \\
\mathbb{E}\left[\mathbf{o}_1\right] &= 1.6 \\
\mathbb{E}\left[\mathbf{o}_2 \mid \mathbf{o}_1\right] &= 0 \\
\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} = 0 < 0.048 &= \Delta \left(\sum_{k=t+1}^{T} \mathbb{E}\left[\mathbf{o}_k \mid \mathbf{o}_{0:t}\right] + \sum_{k=1}^{t} \mathbb{E}\left[\mathbf{o}_k \mid \mathbf{o}_{0:k-1}\right] \right).
\end{aligned}$$

so action a_1^1 is permissible. Note that $\mathbb{E}\left[\mathbf{o}_1\right]$ averages the possible values of $\mathbf{o}_1$, ignoring the actual observation $\mathbf{o}_1$ that as received. On the other hand, for the safe state history $\langle s_0, a_0, s_1, a_1^2, s_2 \rangle$ we find

$$\begin{aligned}
\sum_{t=1}^{T} p_f(s_t) &= 0.1 \\
\mathbb{E}\left[\mathbf{o}_1\right] &= 1.6 \\
\mathbb{E}\left[\mathbf{o}_2 \mid \mathbf{o}_1\right] &= 0 \\
\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} = 0.\bar{1} > 0.048 &= \Delta \left(\sum_{k=t+1}^{T} \mathbb{E}\left[\mathbf{o}_k \mid \mathbf{o}_{0:t}\right] + \sum_{k=1}^{t} \mathbb{E}\left[\mathbf{o}_k \mid \mathbf{o}_{0:k-1}\right] \right).
\end{aligned}$$

so that action a_1^2 is not permissible, and action a_1^1 is selected.

On the other hand, let us say that the agent observed $\mathbf{o}_1 = 1$ in the motivating

example. For the safe state history $\langle s_0, a_0, s_1, a_1^2, s_2 \rangle$, we have

$$\sum_{t=1}^{T} p_f(s_t) = 0.1$$
$$\mathbb{E}\left[\mathbf{o}_1\right] = 1.6$$
$$\mathbb{E}\left[\mathbf{o}_2 \mid \mathbf{o}_1\right] = 2.\bar{1}$$
$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} = 0.\bar{1} < 0.111\bar{3} = \Delta\left(\sum_{k=t+1}^{T} \mathbb{E}\left[\mathbf{o}_k \mid \mathbf{o}_{0:t}\right] + \sum_{k=1}^{t} \mathbb{E}\left[\mathbf{o}_k \mid \mathbf{o}_{0:k-1}\right]\right).$$

so the conditions above is valid, and as a result, action a_1^2 is selected as the highest reward action.

For completeness, we can also consider the case where $\mathbf{o}_1 = 3$. For the safe state history $\langle s_0, a_0, s_1, a_1^2, s_2 \rangle$, we have

$$\sum_{t=1}^{T} p_f(s_t) = 0.1$$
$$\mathbb{E}\left[\mathbf{o}_1\right] = 1.6$$
$$\mathbb{E}\left[\mathbf{o}_2 \mid \mathbf{o}_1\right] = 2.\bar{5}$$
$$\frac{\sum_{t=1}^{T} p_f(s_t)}{1 - \sum_{t=1}^{T} p_f(s_t)} = 0.\bar{1} < 0.124\bar{6} = \Delta\left(\sum_{k=t+1}^{T} \mathbb{E}\left[\mathbf{o}_k \mid \mathbf{o}_{0:t}\right] + \sum_{k=1}^{t} \mathbb{E}\left[\mathbf{o}_k \mid \mathbf{o}_{0:k-1}\right]\right).$$

so the conditions above is valid, and as a result, action a_1^2 is selected as the highest reward action. We have therefore recovered execution of the optimal policy in the conditional CCMDP with our online plans, which we have already verified satisfies the risk bounding function.

4.10.4 Guarantees on Existence of Policy

The above strategy guarantees that a policy can be executed that follows the risk bounding function, assuming that a plan that satisfies (4.30) can be computed online in response to all measurements. This is a non-trivial assertion because a solution that satisfies the constraint may not exist, and in this section we introduce an additional

condition that ensures this is always possible.

Consider planning online from state s_t, with a plan that satisfies (4.30) and includes a_t. Define a *worst case state history* $w(a_t)$ as any safe state history $w(a_t) = \left\langle s_0^{w(a_t)}, a_0^{w(a_t)}, s_1^{w(a_t)}, \ldots, s_n^{w(a_t)} \right\rangle$ that satisfies $s_{0:t}^{w(a_t)} = s_{0:t}$, $a_{0:t}^{w(a_t)} = a_{0:t}$, and

$$\frac{\sum_{t=1}^{T} p_f(s_t^{w(a_t)})}{1 - \sum_{t=1}^{T} p_f(s_t^{w(a_t)})} \leq \Delta \left(\sum_{k=1}^{t+1} \mathbb{E}\left[\tilde{g}_k(\mathbf{o}_{0:k}) \mid \mathbf{o}_{0:k-1} \right] \right). \tag{4.33}$$

By requiring that any worst case state history exists for the chosen a_0 from the initial state s_0, the fact that a solution always exists to all online CCMDPs in this chapter is guaranteed.

To see this, assume that when planning from state s_t that a plan is found that satisfies (4.30) and $w(a_t)$ exists. $w(a_t)$ is intuitively a plan that satisfies Vulcan's condition, even without any additional reward from future observations. When planning online from any possible s_{t+1}, after observing any $\mathbf{o}_{t+1}$, following the actions $a_{t+1:T-1}^{w(a_t)}$ must therefore satisfy (4.30) as well, even if all measurements result in zero expected reward.

The actions $a_{t+1:T-1}^{w(a_t)}$ from any s_{t+1} is also a worst case state history for $a_{t+1}^{w(a_t)}$, which means that the worst case state history could be followed until the end of execution, guaranteeing that (4.30) and (4.33) can always be satisfied regardless of measurements.

When then planning online from s_{t+1}, usually the action selected $a_{t+1} \neq a_{t+1}^{w(a_t)}$. In this case, a_{t+1} is only permitted if $w(a_{t+1})$ can be found, which guarantees the risk bounding function can be satisfied regardless of future measurements from a_{t+1} by the same argument.

To summarize, the worst case state history is typically not executed, but we re quire that one exists. This is a weaker condition than enforcing that the solution to the online CCMDP must *be* a worst case state history. Furthermore, the existence of a worst case state history implies a worst case state history exists when planning after the next measurement. So long as a worst case state history exists at state s_0, one necessarily exists for all s_t, and so a solution to all online CCMDPs necessarily

exist.

Practically, the existence of a worst case state history can be checked after sampling a_t when planning from s_t. If one cannot be found, the action is immediately deleted from the search space. Worst case state histories can often be found by greedily selecting the minimum risk action from the action space. In our implementation, if this is not a valid worst case state history, none is assumed to exist.

4.11 Algorithm Description

Our strategy requires solving T online CCMDPs, and we provide an algorithmic description in this section. Algorithm 13 provides an overview of online execution using online CCMDPs. Algorithm 13 operates on an agent described through its initial observations and position, and takes a total planning horizon T, plus a planning time limit τ to use when solving each online CCMDP. The current state of the agent, on line 2, captures the observations so far, the mean and covariances of positions so far, and that the current state is a safe state. If the current state were not safe, planning would not be proceeding. The T online CCMDPs are constructed and solved in each of the loops on line 3. The best actions in the online CCMDPs are executed, and the next observation $\mathbf{o}_{t+1}$ is observed if the action successful.

Algorithm 13: ExecuteRiskBoundedPolicy

Input: Planning horizon T, planning time limit τ, initial observations $\mathbf{o}_0$, initial agent position distribution $\boldsymbol{\mu}_0$, $\boldsymbol{\Sigma}_0$

```
1 for t from 0 to T − 1
2 |   s_t ← state with o_{0:t}, μ_{0:t}, Σ_{0:t}, and F_t = 0
3 |   a ← RiskBoundedMCTS(s_t, T, τ)
4 |   Execute a
5 |   if a results in safe state then
6 |   |   observe o_{t+1}
7 |   else
8 |   |   end mission
```

Algorithm 14 describes the MCTS strategy for solving the online CCMDPs. It operates on a state s_t at time step t, and requires a planning horizon T and a time to

use for planning τ. Each online CCMDP is solved by repeatedly performing rollouts from s_t up to a time limit of τ, in the loop on line 1. A search tree is built from s_t according to the action selection rules of the UCT algorithm.

Lines 5 through 15 provide rules for deleting actions in search tree, so that the plan found from s_t satisfies Vulcan's conditions for chance constraint satisfaction. Upon reaching the planning horizon, (4.30) is verified in the if statement on line 5, and the previous action is deleted if it is not satisfied. Similarly, if no actions remain at a state after all actions leading to it are deleted on line 8, then the previous action is deleted. On line 12, after selecting action a_t, $w(a_t)$ is found and (4.33) is verified. If this condition fails, the immediately preceding action is also deleted.

Lines 16 through 21 define immediate rewards for states. These are assigned at leaf states, regardless of whether they result in failure. Other states receive zero immediate reward.

The rollout continues on line 22, with actions selected according to UCT's selection rule, and child states constructed from each action. In the online CCMDPs constructed here, the states represent all observations that could occur, or failure. A failure state is generated if a random number is below the probability of failure from the selected action a, otherwise a success state is generated. The next state is sampled, returning the reward achieved from all states below on the rollout. None could be returned if the rollout resulted in actions being deleted by violation of the constraints we impose. If it is note, the estimate for $\hat{Q}$ is updated at the state, and counts in the number of times samples are taken are incremented.

4.12 Experiments

We examine our algorithm in two different ways. First, we run the algorithm on real bathymetry data and a simulated measurement field. We show our algorithm is able to move towards high reward locations based on the data it gathers, and take dangerous actions when they are expected to yield high reward. We then verify that the risk bounding function is satisfied through Monte Carlo simulations over randomly

Algorithm 14: RiskBoundedMCTS

Input : State s_t, planning horizon T, max planning time τ
Output: Next action to execute

loop until planning time τ reached
  `SampleState(`s_t`)`
return $\arg\max_a \hat{Q}(s_t, a)$
Procedure `SampleState(`$s_{t'}$`)`
  if $t' = T$, $s_{t'}$ is a safe state, and (4.30) not satisfied **then**
    delete $a_{t'-1}$
    return None
  if no actions remain at $s_{t'}$ **then**
    delete $a_{t'-1}$
    return None
  if $t' = t + 1$ **then**
    $w \leftarrow$ greedily found sequence of min risk states
    if (4.33) not satisfied **then**
      delete a_t
      return
  if $s_{t'}$ is a failure state **then**
    return $\mathbb{E}\left[g(\mathbf{o}_{t+1:t'} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right]$
  else if $t' = T$ **then**
    $r \leftarrow \mathbb{E}\left[g(\mathbf{o}_{t+1:T} \mid \mathbf{o}_{0:t}) \mid \mathbf{o}_{0:t}\right]$
  else
    $r \leftarrow 0$
  if $t' \neq T$ **then**
    $a \leftarrow \arg\max_{a_{t'}} \hat{Q}(s_{t'}, a_{t'}) + \sqrt{\frac{2 \log N_{s_{t'}}}{N_{s_{t'}, a_{t'}}}}$
    if `Random(0, 1)` $< p_f(s_{t'+1})$ **then**
      $s_{t'+1} \leftarrow$ child state with $\boldsymbol{x}_{t'+1} = \boldsymbol{x}_{t'} + d(a)$, $\boldsymbol{\Sigma}_{t'+1} = \boldsymbol{\Sigma}_{t'} + \boldsymbol{\Sigma}_w$, $F_{t'+1} = 1$
    else
      $s_{t'+1} \leftarrow$ child state with $\boldsymbol{x}_{t'+1} = \boldsymbol{x}_{t'} + d(a)$, $\boldsymbol{\Sigma}_{t'+1} = \boldsymbol{\Sigma}_{t'} + \boldsymbol{\Sigma}_w$, $F_{t'+1} = 0$ representing all $\mathbf{o}_{t'+1}$
    $q \leftarrow$ `SampleState(`$s_{t'+1}$`)`
    if q is not None **then**
      $\hat{Q}(s_{t'}, a) \leftarrow \frac{N_{s_{t'}, a} \ \hat{Q}(s_{t'}, a) + q}{N_{s_{t'}, a} + 1}$
      $N_{s_{t'}, a} \leftarrow N_{s_{t'}, a} + 1$
      $r \leftarrow r + q$
      $N_{s_{t'}} \leftarrow N_{s_{t'}} + 1$
  return r

instantiated Gaussian Processes.

To test the performance of our algorithm in realistic scenarios, we convexify true bathymetric data to produce forbidden regions, and simulate a temperature measurement field. The location chosen was East of Boston Harbor, from -70.890 to -70.876 degrees longitude, and 42.344 to 42.355 degrees latitude, provided by NOAA survey H10992 [97]. The mission simulated an autonomous underwater vehicle operating at a constant 15 meters depth, with an objective to maximize the sum of its temperature measurements, which are taken with negligible noise.

15 meter depth contours were used as obstacle boundaries. In each case, the agent started at a location -70.8816 degrees longitude and 42.3505 degrees latitude with zero position uncertainty. The environment was modeled as a single attribute Gaussian process with a radial basis function kernel. The true value of the measured field was 16 at the starting location, and increased by 1 for each km West or South. Each action moved the vehicle 50 meters in one of the eight compass directions. We used the following parameters: $T = 20$, $\tau = 60$ sec, $\mathbf{\Sigma}_0 = 0I$ m^2, $\mathbf{\Sigma}_w = 12I$ m^2, $m(x) = 16$, $k(x, x') = 1.25 \exp\left\{\left(-\left\|x - x'\right\|^2 / (2 \times (200 \text{ m})^2)\right)\right\}$.

In this experiment, the Gaussian process environment model meant the observations could be arbitrarily low, and this makes it difficult to construct a concave risk bounding function that is valid over all possible values of the reward. Furthermore, we wish to avoid missions where the agent repeatedly samples a single location that is known to be high. In order to avoid handling very low rewards and repeatedly sampling the same state, the objective that the agent maximizes is set to be

$$\mathbb{E}\left[\sum_{t=1}^{T}(\mathbf{o}_t - 12.5)\ \mathbb{1}\{\mathbf{o}_t \geq 12.5, \|\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t'}\| > 12.5 \text{ m } \forall t' < t\} \mid \mathbf{o}_0\right],$$

meaning that observations below 12.5, or taken at locations with a mean location within 12.5 meters of a previous observation are assigned zero reward, otherwise the reward is the observation minus 12.5.

4.12.1 Tests on Controlled Environments

In Figure 4-6 we test this scenario using three different risk bounding functions. The risk bounding functions were selected primarily to show differences in behavior, but we note that they lead to realistic acceptable failure rates on the order of tenths of a percent. Figure 4-6 (a) shows a trajectory resulting from a risk bounding function $\Delta(x) = 0.0003x$. The measurements are high enough to warrant movement into the South-West of the map by the most direct route possible, which requires passing between multiple obstacles. In Figure 4-6 (b) a lower risk bounding function of $\Delta(x) = 0.0002x$ does not allow movement close to obstacles until there is enough certainty that high measurements lie to the South-West. In addition, the route taken uses a thicker channel, with less overall probability of failure. Finally, in Figure 4-6 (c), the risk bounding function $\Delta(x) = 0.0001x$ is too strict to allow the vehicle to pass close to obstacles, and instead it moves up and down the border of the obstacles without moving in too close.

4.12.2 Monte Carlo Tests

In order to experimentally verify that the risk bounding function is satisfied across a policy, we ran Monte Carlo simulations with random measurements following a known Gaussian Process, and verified that the failure rate was less than the risk bounding function applied to the average reward. In the simulations, true (disturbed) locations were generated, and measurements were drawn from a Gaussian Process at the true locations, while the algorithm reasoned over measurements at mean locations. The environment was the same as the previous test.

The low probability of failure in the previous test meant that uncertainty in a Monte Carlo derived failure rate would be comparable to the true failure rate. To increase certainty in the simulation results, we increased the covariance of the agent, leading to a higher probability of failure. In order to speed up simulations and emphasize risk, we also decreased the planning horizon and planning time, and allowed the agent to move further with every action. The following parameters were changed:

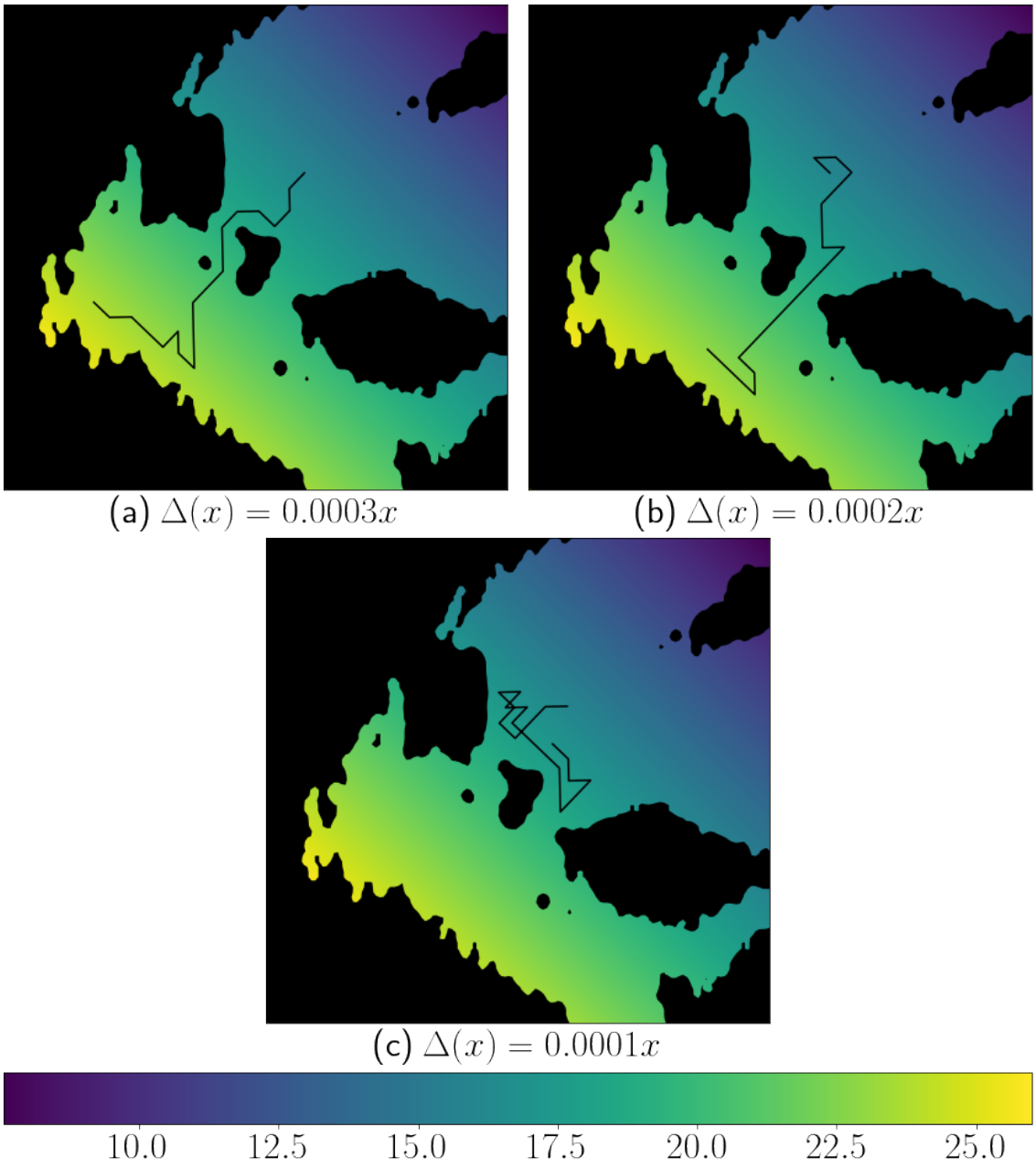


Figure 4-6: Output trajectories with a single true environment and multiple risk bounding functions.

$n = 8$, $\tau = 2$ sec, $\mathbf{\Sigma}_w = 60I$ m^2, $l_{min} = 25$ m. The distance traveled by every action was changed to 100 m. Failure was evaluated with respect to the convexified obstacles, so the failure rate does not account for conservatism due to convexification. 10,000 simulations were run with a risk bounding function of $\Delta(x) = 0.001x$. The mean function was the true environment of the previous experiment so that measurements would typically be biased towards dangerous actions.

The expected cumulative reward was 64.9, which permitted a failure rate of 0.0649 under the risk bounding function, while the measured failure rate was 0.0208. There was conservatism in the policy, as only 32% of permitted risk was used. The conservatism can be attributed to three major sources. First and most importantly, our strategy averages reward and risk across outcomes, but does not move all allowed probability of failure from low risk to high risk outcomes. In particular, some environments resulting from the GP have high reward to the East where there are few obstacles. Our approach is not fully capable of moving all allowed risk to cases where high rewards are near obstacles. The additional sources of conservatism are the use of Boole's inequality to overestimate the probability of failure and the underestimation of the reward function.

To confirm that conservatism was reduced when danger exists in all directions, we reran the experiment with $\mathbf{\Sigma}_w = 100I$ m^2 and additional obstacles introduced to the North and East. In this case the expected cumulative reward was 26.4, which permitted a failure rate of 0.0264, while the measured failure rate was 0.0165. In this case, 63% of available risk was used.

4.13 Summary

In this chapter, we developed a method of executing a mission where the probability of failure is bounded as a convex function of expected reward. By applying Monte Carlo tree search to a series of easily computable online problems, we ensure that an action is found in an anytime manner. We derived constraints that enforce a chance constraint without the need to plan over all outcomes and which guarantee online planning is

possible. Simulation results on true bathymetry show our algorithm trades off risk against reward intuitively, taking dangerous actions only when justified by the reward, while Monte Carlo simulations verify that the chance constraint is satisfied.

Chapter 5

The Directed Acyclic Gaussian Process (AcyGP) Model

Query-driven adaptive sampling is performed in environments with complex interdependencies between observable variables. For example, the probability of presence of a hydrocarbon seep at a location is strongly dependent upon whether other bathymetric features like mounds or pockmarks are present at that location. In order to generate accurate probability distributions over variables at unobserved locations, we desire an environment model that will allow those relationships to be modeled, and learned when they are unknown. In order to allow experts to encode known relationships between variables, the interdependencies should be expressed in an intuitive manner.

In this chapter, we propose the AcyGP model as the environment prediction model for query-driven adaptive sampling. The AcyGP model extends existing multi-output Gaussian process methods by encoding the relationships between different attributes through a directed acyclic graph. In this way, the AcyGP model combines the capability of directed acyclic graphs to perform accurate prediction with limited data, with the capability of Gaussian processes to capture spatio-temporal correlations.

In the AcyGP model, the relationships between attributes are simple and interpretable, which allows an expert user to specify qualitative relationships that must appear in the model, or exclude certain relationships from consideration. Unknown

dependencies between variables can be automatically discovered from data, by a search procedure that finds the structure that best fits the data. We show that by capturing the dependency structure of the environment, the AcyGP model outperforms existing multi-output Gaussian process models on a variety of prediction domains. In addition, we show that the learned structure has a physically meaningful interpretation.

5.1 Motivation

One of the key principles of any adaptive sampling procedure is that variables in the environment are correlated. These correlations allow the sampling agent to identify promising places to observe, based on the probabilistic model learned thus far. The correlations include spatio-temporal correlations and correlations between two different attributes. An example spatial correlation is that a hydrocarbon seep is much more likely to be present within a few hundred meters of where another has already been observed. A temporal correlation could be that at a fixed location, a seep is more likely to be actively releasing gas when it has recently been observed to be active. Correlations between two attributes may include that a seep is much more likely to be observed at a location where an elevated backscatter signal has been observed, or where bathymetric features like mounds have been found. The presence of such correlations allows an adaptive sampling procedure to update predictions of unobserved variables using nearby or related observations. The updated predictions are then used to decide where an adaptive sampling algorithm should take additional measurements.

Gaussian processes (GPs) are frequently used for environment models in adaptive sampling for two key reasons. First, GPs are able to encode correlations in space and time, and between multiple environmental attributes, thereby modeling all correlations of interest in an environment. Second, GPs produce distributions as predictions, making them suitable for the probabilistic reasoning in adaptive sampling. Unlike many other probabilistic machine learning methods, the predictions

from GPs are consistent with Bayesian inference, in the sense that prediction uncertainty decreases on average once additional data has been added.[1] This means that it is actually meaningful to compute the information gained by an observation in a GP, because the predictions behave as a random variable should in response to new data.

Multi-output Gaussian processes (MOGPs) model multiple attributes, such as the presence of seepage and temperature, throughout space and time. In the machine learning community, each attribute is known as an *output*, because it is output by the Gaussian process model. In this chapter, we will mostly use the term attribute, but we will continue to use the acronym MOGP to maintain consistency with the Gaussian process literature.

Each attribute exists at each *input*, which is typically a location in space and/or a point in time. Correlations are learned between variables of the same attribute at different inputs, and between separate attributes. Parameterizations that encode the correlations, such as length scales of spatial correlations, are trained on the observed data, in order to best learn patterns that appear in that data. Correlations between attributes allow data from one attribute to be used to improve prediction accuracy in another.

Existing MOGP approaches are effective at learning complex correlations between attributes, but many state of the art models learn correlations between all pairs of attributes [4, 54, 110, 132]. This is undesirable because in complex adaptive sampling problems not every attribute is necessarily correlated or relevant for prediction of variables that impact a query. Relatively little work in MOGPs has focused on determining whether there is sufficient evidence for two attributes to be correlated. By correlating all attributes, MOGP training may capture random correlations resulting from small sample statistics between independent attributes. This results in less ac-

[1] A regression algorithm can be made to output a predictive probability distribution by predicting parameters of the distribution. A common way to do this is to train a neural network to output the mean and variance of a Gaussian distribution. However, since the neural network does not treat the prediction explicitly as a random variable, there is no guarantee that predictions will satisfy laws of conditioning. A prior prediction may not be equal to the average of posterior prediction that is conditioned on more data, and the average uncertainty of posterior predictions may actually increase, dependent upon how the network has been parameterized.

curate predictions, since a change in one attributes will lead to changes in another that is meant to be unrelated. Correlating all attributes will also directly correlate two attributes that are independent when conditioned on a third, resulting in changes in predictions even when the third attribute does not change, and again lowering accuracy. In these cases, the false correlations cause an MOGP to duplicate patterns from one attribute to another, even when there is limited direct evidence that the same pattern exists in both attributes. This leads to poor predictive performance and underestimation of prediction errors.

Another drawback of existing MOGP models is their inflexibility. In query-driven adaptive sampling, we aim to improve prediction accuracy in the face of limited data by using expert knowledge, including textbook knowledge and prior experience, to guide the model. Experts can do so by specifying which correlations between attributes do or do not exist, as well as qualitative properties of these relationships, such as positive and negative correlations. Existing fully correlated MOGP methods provide no simple method to remove correlations between specific pairs of attributes or impose qualitative knowledge.

Some methods have been reported that control the attributes that are correlated, by manually specifying all pairs of attributes that are correlated [1, 70, 83]. In these methods, predictions are highly dependent on the structure choice. These methods work well when the full structure is known, but even an expert may not be able to exactly specify all correlations that do or do not exist. With the AcyGP model, we allow the user to specify restrictions on the space of models. When an expert knows certain correlations exist or do not exist, they can specify this knowledge as a constraint. Likewise, when qualitative relationships are known, they can be encoded. Bayesian networks, which are directed acyclic graphs for modeling conditional probabilities, provide a framework to naturally model these kind of relationships, which we combine with a GP model to describe the spatial correlations in environments. For all correlations and relationships that are not specified, the model is trained on data to recover the most likely sets of correlations consistent with the user's restrictions.

Control of correlations between variables has been considered extensively in the

graphical models community [67, 73]. As with fully connected multi-output GPs, fully connected graphical models tend to propagate noise or correlate conditionally independent attributes, reducing predictive performance [43]. Structure learning algorithms have been developed in the graphical models community to identify simple, high likelihood models that enforce likely conditional independence statements between variables, and can be made consistent with restrictions on correlations that must exist. The graphical models found through structure learning improve predictive accuracy in the presence of conditionally independent attributes, and are more robust to noise.

In this chapter, we leverage insights from structure learning of graphical models to introduce a Gaussian process model called the AcyGP model. The AcyGP model combines latent GPs in a directed acyclic graph (DAG) structure, in order to control attribute correlations. The DAG structure provides an intuitive method of specifying those correlations, because the connections are directly between attributes and not through alternative hidden processes. The structure also enables a degree of interpretability in the model, because the edge relationships each rely on few parameters, and the effect of each on the model can be understood. The DAG may be partially specified by an expert, and the remainder is learned from the data. When learning the unspecified structure, we limit it to only those conditional independencies between attributes that are substantially justified by the data.

5.2 Related work

Gaussian processes model data at various inputs as Gaussian distributed. The covariance between data points is modeled as a function of the relative inputs of different data points, so observations at inputs that are closer together are more closely correlated. A Gaussian process is trained by optimizing a parametric representation of the covariance matrix, called the covariance kernel, in order to optimize the likelihood of the data [118, 147].

A frequently taken approach to multi-output Gaussian process regression is to

combine independent single attribute Gaussian processes to produce multiple correlated attributes. The independent Gaussian processes are unobserved, and so are considered to be latent processes in the model. The observed attributes are modeled by linearly combining the latent processes with different weights for each attribute, or by convolving the latent processes [5, 87].

Linear combination models including the intrinsic coregionalization model [68], the linear model of coregionalization [54], and the semiparametric latent factor model (SLFM) [132], all represent attributes as linear sums of the latent processes. This means an attribute at a certain input only depends on the latent processes at that same input, making these approaches simple to construct. The methods differ in the number of latent processes used and whether the latent processes share kernels. Sharing latent processes between attributes leads to correlations between attributes, and the fact that each latent process may have correlations over a different length scale leads to correlations over multiple spatial scales for each attribute. Extensions have explored applying higher rank combinations of the latent processes [18], allowing correlated latent processes [139], and adding an additional latent process unique to each attribute [98] to construct more expressive models. In all cases, predictive performance is strongly dependent upon choices of the number of latent processes and how they are mixed. Determination of the optimal model requires training all possibilities and testing. In contrast, our use of structure learning may be viewed as an automated method of selecting which latent processes contribute to which attribute variables.

Convolutional methods attempt to integrate 'non-local' information, so that an attribute at a certain input may depend on the values of latent processes at different inputs. Convolutional methods construct attribute processes by convolving latent processes with a smoothing kernel, incorporating information from multiple inputs. Initial approaches used a single white noise process [141, 94], while more recent extensions have used mixtures of multiple latent processes [19] and allowed latent processes different from white noise [3, 81]. Convolution allows non-local effects such as time delays to be modeled, but this leads to more hyperparameters to train compared to

linear models. This results in a greater potential for noise transfer between attributes and additional predictive error if the number of latent processes is not carefully controlled.

The AcyGP method is most similar to a class of methods we refer to as Gaussian process autoregressors, where attributes are considered sequentially and are computed using transformations of previous attributes. Dependent on the choice of transformation, this can be more expressive than linear combinations of latent processes. The sequence of transformations leads to a directed graph structure, where each attribute is a child of those of which it was a transformation. Work on GP autoregressors, to date, has used a fully connected network [110], a single parent for each attribute [70], or a bipartite network [83]. Choices among these three structures are application driven, designed to correlate attributes according to known relationships.

Our method has the same goal as GP autoregressors, of controlling correlations between attributes, but we generalize on these methods by allowing any DAG structure between attributes by learning that structure directly from the data. The possible structures that our method automatically learns includes those used in the previously mentioned work, but typically the best structure found by our method falls outside these three categories.

Gaussian processes autoregressors also bear similarity to deep Gaussian processes [34]. Deep GPs use the attributes of latent GPs as inputs to another layer of GPs, rather than networks between attribute dimensions. These models are highly expressive, but are difficult to analyze and train.

Finally, a related body of work has considered inference on Gaussian processes defined over inputs that are connected by trees and general undirected graphs [130, 145, 140]. These methods improve prediction accruacy when using small training sets. In these cases the graphical model is imposed over the input space for GPs with a single attribute variable and does not control correlations between multiple attributes, whereas we consider a graphical model imposed between multiple attribute variables.

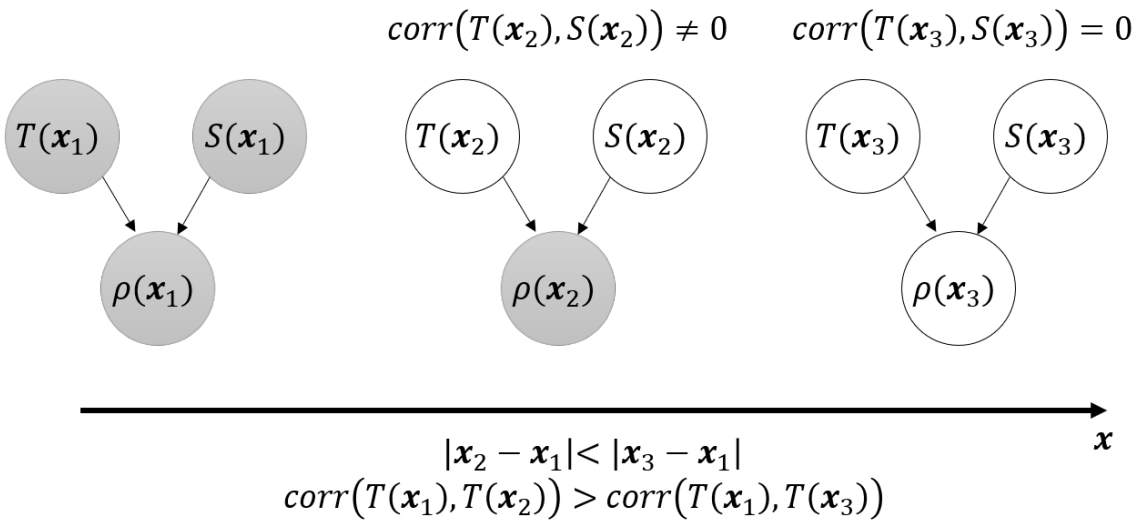


Figure 5-1: Three oceanic attributes modeled as a function of $\boldsymbol{x}$. Gray nodes are observed, white nodes are unobserved. Correlation decays with distance. When $\rho(\boldsymbol{x}_2)$ is observed, $T(\boldsymbol{x}_2)$ and $S(\boldsymbol{x}_2)$ are correlated. When $\rho(\boldsymbol{x}_3)$ is not observed, $T(\boldsymbol{x}_3)$ and $S(\boldsymbol{x}_3)$ are uncorrelated

5.3 Overview of the AcyGP Model

In this chapter, we describe the AcyGP model, which we claim removes spurious correlations that are introduced by limited data, and allows an expert to encode their knowledge into the environment model. The AcyGP model does this by constructing a multi-output Gaussian process from single attribute Gaussian processes that are arranged in a directed acyclic graph structure. The AcyGP model makes dependencies between attributes explicit, so that qualitative expert knowledge can be converted into constraints.

To motivate the development of the AcyGP model, let us consider an environment in which we model three oceanographic attributes; temperature T, salinity S, and density ρ. Every point $\boldsymbol{x}$ in the environment has a temperature, salinity and density, so we model these as functions $T(\boldsymbol{x})$, $S(\boldsymbol{x})$, $\rho(\boldsymbol{x})$, as shown in Figure 5-1. When we have some observations of these variable at a location $\boldsymbol{x}_1$, and partial observations at $\boldsymbol{x}_2$, and we want to be able to predict them at other locations $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$.

We may not know the attributes at $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$, but there are various chains of logic that would guide our predictions, and that we want our model to capture. The

first is the idea of spatial correlation. If $\boldsymbol{x}_2$ is close to $\boldsymbol{x}_1$, then the temperature at $\boldsymbol{x}_1$ is likely similar to the temperature at $\boldsymbol{x}_2$. The temperature at $\boldsymbol{x}_3$ that is further away is more likely to be have a larger difference from $T(\boldsymbol{x}_1)$. The same logic applies to $S(\boldsymbol{x})$ and $\rho(\boldsymbol{x})$, though each may have different strengths of spatial correlations. The idea of spatial correlation is captured by a Gaussian process, which is able to produce a predictive distribution for $T(\boldsymbol{x}_2)$ and $T(\boldsymbol{x}_3)$.

The second idea that guides our predictions is that attributes are correlated with each other. Greater densities are observed at greater salinities and lower densities are observed at higher temperatures. If we had observed $\rho(\boldsymbol{x}_2)$ to be higher than $\rho(\boldsymbol{x}_1)$, then we might expect that $S(\boldsymbol{x}_2) > S(\boldsymbol{x_1})$ and $T(\boldsymbol{x}_2) < T(\boldsymbol{x}_1)$, though we would not know for sure since other unobserved variables like pressure may also contribute to changes in density. Multi-output Gaussian processes capture this knowledge by learning correlations between the attributes, so that they can inform predictions.

But there is even more knowledge available to experts that is not captured by MOGP models. An expert may know that increasing the temperature of a packet of water will cause it to expand, reducing density. Increasing salinity means adding more mass, also causing an immediate change in density. This is a causal relationship, indicating that density is directly affected by changes to temperature and salinity. This causality implies that temperature and salinity are correlated in specific ways when density is observed. If after observing that $\rho(\boldsymbol{x}_2) > \rho(\boldsymbol{x}_1)$ we want to predict $T(\boldsymbol{x}_2)$ and $S(\boldsymbol{x}_2)$, then it is likely that either $T(\boldsymbol{x}_2) < T(\boldsymbol{x}_1)$ or that $T(\boldsymbol{x}_2) > T(\boldsymbol{x}_1)$ and $S(\boldsymbol{x}_2) > S(\boldsymbol{x}_1)$. Temperature and salinity are dependent when conditioned on density. Moreover, we know the direction of these relationships already, and the environment model should be able to capture and encode that knowledge that we already know, even if there is not enough data to learn it.

Without observing changes in density, changes in temperature and salinity can be explained by a variety of influences like tributaries and sunshine, and they can behave effectively independently to each other. This is the idea of conditional independence that is also not captured by existing MOGP models. Existing MOGPs would learn correlations between $T(\boldsymbol{x})$ and $S(\boldsymbol{x})$, even when they are only correlated through

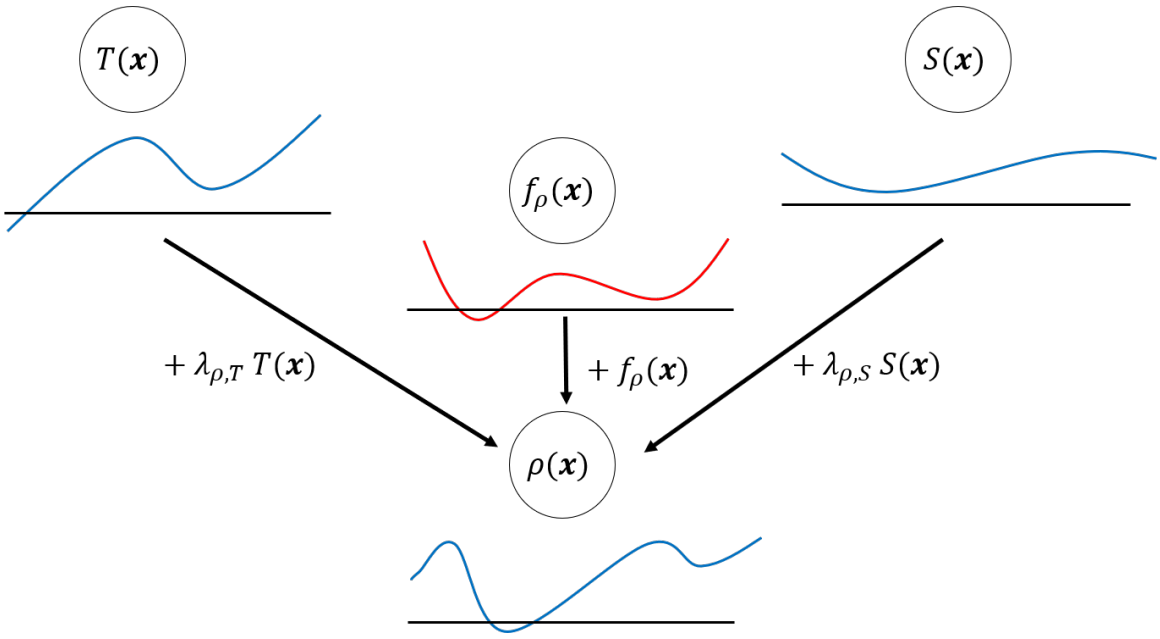


Figure 5-2: Sketch of a AcyGP constructed for three continuous unbounded variables. Blue functions represent modeled attributes, while the red line indicates a latent process. Three Gaussian processes are combined in a DAG structure to model the environment.

$\rho(\boldsymbol{x})$. Learning this correlation constrains predictions of $T(\boldsymbol{x})$ and $S(\boldsymbol{x})$, and reduces accuracy.

The AcyGP model captures all this information using a directed acyclic graph structure. A schematic of the structure of a basic AcyGP is given in Figure 5-2. We model $T(\boldsymbol{x})$ and $S(\boldsymbol{x})$ as Gaussian processes that are independent of one another, since they do not depend on one anther except through $\rho(\boldsymbol{x})$. Since $\rho(\boldsymbol{x})$ is affected causally by $T(\boldsymbol{x})$ and $S(\boldsymbol{x})$, we construct $\rho(\boldsymbol{x})$ as a linear combination of $T(\boldsymbol{x})$, $S(\boldsymbol{x})$, and an unobserved (latent) process $f_\rho(\boldsymbol{x})$. $f_\rho(\boldsymbol{x})$ encodes changes in density that are not fully explained by temperature and salinity, and may have its own spatial correlations.

This construction, where density is dependent upon temperature and salinity, defines a directed graphical model, in which there are edges from $T(\boldsymbol{x})$ to $\rho(\boldsymbol{x})$ and $S(\boldsymbol{x})$ to $\rho(\boldsymbol{x})$. $T(\boldsymbol{x})$ and $S(\boldsymbol{x})$ do not depend directly on one another, so there is no edge between them, and they are not transformations of each other. In this way, the AcyGP model captures conditional independence between attributes. Since $\rho(\boldsymbol{x})$ is

constructed using linear multiples of $T(\boldsymbol{x})$ and $S(\boldsymbol{x})$, we can encode our qualitative knowledge of these relationships as constraints on the coefficients $\lambda_{\rho,T}$ and $\lambda_{\rho,S}$ in the model. Knowing that density decreases as temperature increases means that $\lambda_{\rho,T} < 0$, and knowing that density increases as salinity increases means that $\lambda_{\rho,S} > 0$. In order to train the AcyGP, we learn the structure and the parameters, but an expert can specify the edges that must exist, and specify the constraints on parameters when training begins.

The model we described so far has all unbounded, continuous attributes, which makes modeling simple. But in adaptive sampling, we may want to model more complex attributes. For example, we may wish to model the presence of mounds, the presence of elevated backscatter, and the presence of seeps as $M_p(\boldsymbol{x})$, $B_p(\boldsymbol{x})$, and $S_p(\boldsymbol{x})$ respectively. These are functions that at each point in space are binary. Let us say that we want to capture a similar relationship between attributes, so that seeps depend causally on mounds and backscatter. An example of a AcyGP constructed for this environment is given in Figure 5-3.

Binary attributes are not easily modeled by a Gaussian process, but at a specific location $\boldsymbol{x}$, they can be modeled as a Bernoulli distribution $Bern(\mu(\boldsymbol{x}))$, where $0 \leq \mu(\boldsymbol{x}) \leq 1$ indicates the probability that the attribute is 1 at location $\boldsymbol{x}$. In order to model spatial correlations, we desire that $\mu(\boldsymbol{x}_1)$ and $\mu(\boldsymbol{x}_2)$ are close when $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are close. To do so, we model the parameter $\mu(\boldsymbol{x})$ using a real valued function $\alpha(\boldsymbol{x})$ that can be modeled with a Gaussian process. But since a Gaussian process produces predictions in the range $(-\infty, \infty)$, we compute $\mu(\boldsymbol{x})$ as $e^{\alpha(\boldsymbol{x})}/(1+e^{\alpha(\boldsymbol{x})})$. To model the three attributes, we construct a latent process for each attribute, $f_{M_p}(\boldsymbol{x})$, $f_{B_p}(\boldsymbol{x})$, and $f_{S_p}(\boldsymbol{x})$, which models their individual effects on probabilities of occurrence through space. For mounds and backscatter, $\alpha_{M_p}(\boldsymbol{x}) = f_{M_p}(\boldsymbol{x})$ and $\alpha_{B_p}(\boldsymbol{x}) = f_{B_p}(\boldsymbol{x})$.

In order to introduce dependency between seep presence and mound presence, $\alpha_{S_p}(\boldsymbol{x})$ is composed of a sum of $f_{S_p}(\boldsymbol{x})$ and functions of $M_p(\boldsymbol{x})$ and $S_p(\boldsymbol{x})$. Making these functions linear multiplications would not be very expressive, since $M_p(\boldsymbol{x}) = 0$ would always have no effect on seep presence. Instead, if $M_p(\boldsymbol{x}) = 0$ we add one number, $\lambda_{S_p,M_p,0}$, to $\alpha_{S_p}(\boldsymbol{x})$, and we add a different number $\lambda_{S_p,M_p,1}$ to $\alpha_{S_p}(\boldsymbol{x})$ when

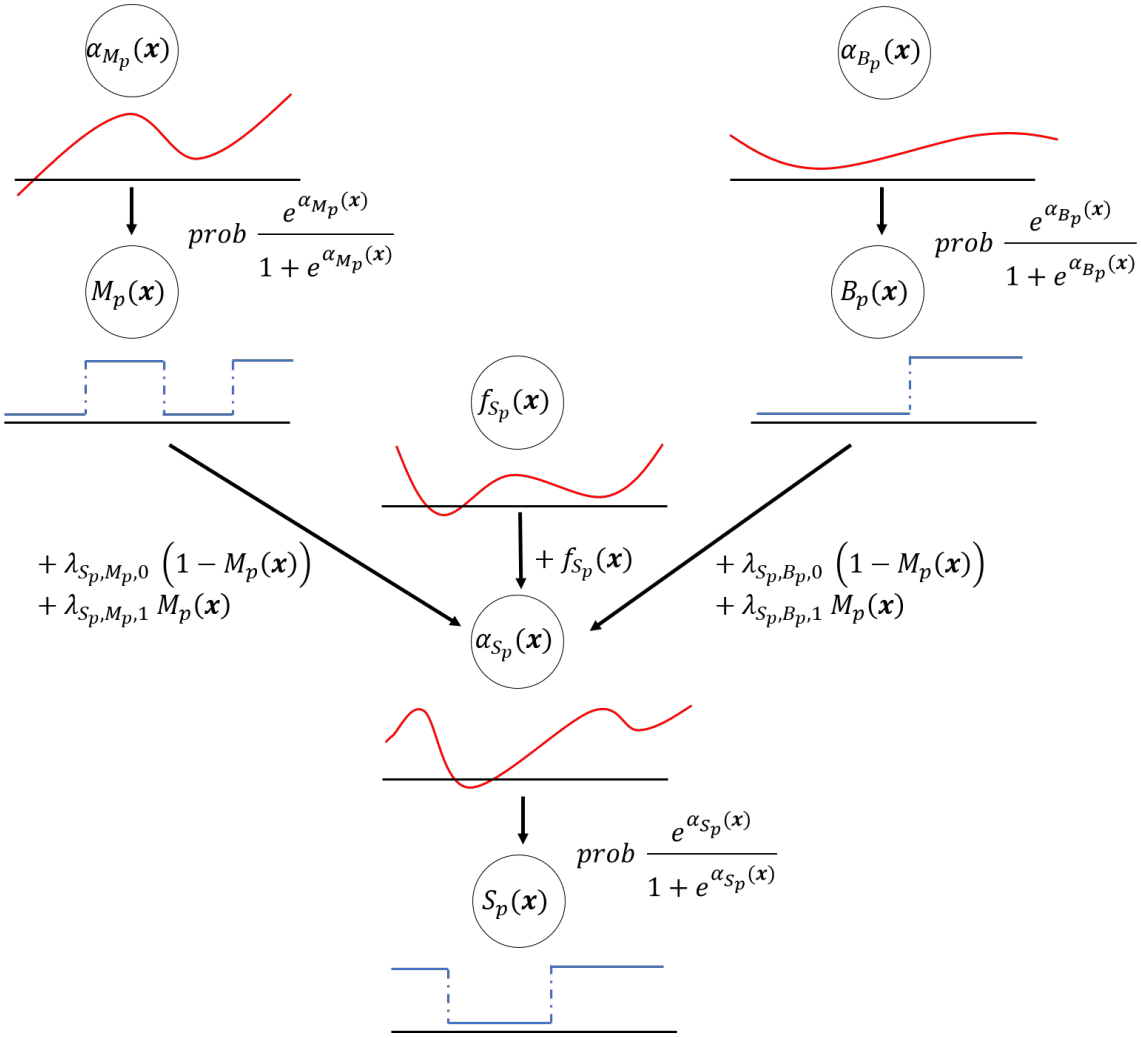


Figure 5-3: Sketch of a AcyGP constructed for three binary variables. Blue functions represent modeled attributes, while red lines indicates latent processes and functions. Three Gaussian processes are combined in a DAG structure, and passed through sigmoid functions to predict binary probabilities, in order to model the environment.

$M_p(\boldsymbol{x}) = 1$. In this way we can scale the relative effects of presence and absence arbitrarily. Training the AcyGP means finding this structure, and solving for the parameters associated with it.

The remainder of this chapter describes this model in full. We provide preliminaries of quadrature rules, Gaussian processes, and directed acyclic graphical models in Sections 5.4, 5.5, and 5.6 that will be necessary for the development of the AcyGP model. We then introduce the homogeneous AcyGP model in Section 5.8, where all variables are continuous and unbounded, because it is a conceptually simpler construction. We discuss how to train the model, and how to produce predictions when the model is trained. We then discuss how qualitative constraints are encoded in the homogeneous AcyGP model in Section 5.9. Finally, we introduce the heterogeneous AcyGP model in Section 5.10, which generalizes the homogeneous AcyGP model and allows for attributes that may be categorical, or bounded. At the end of this chapter, we provide experimental results that show the AcyGP model provides more accurate predictions than existing MOGP methods.

5.4 Preliminaries: Quadrature Rules

In our development of the AcyGP model to model non-continuous attributes, we will require computation of complex expectations that cannot be evaluated in closed form, such as

$$\mathbb{E}_{p(\mathbf{r})}\left[\frac{e^{\mathbf{r}}}{1+e^{\mathbf{r}}}\right] = \int_{\mathcal{R}} p(\mathbf{r})\ \frac{e^{\mathbf{r}}}{1+e^{\mathbf{r}}}\ d\mathbf{r},$$

where $\mathbf{r}$ is Gaussian distributed. Here, $e^{\mathbf{r}}/1+e^{\mathbf{r}}$ is a formula for a probability $p(y \mid \mathbf{r})$, and we will need to evaluate these integrals for many types of probability distributions $p(y \mid \mathbf{r})$. We make use of Gaussian quadrature rules in order to rapidly approximate these expectations without resorting to Monte Carlo techniques.

A C-point quadrature rule for the function $W(\mathbf{r})$ approximates the integral

$\int_{\mathcal{R}} W(\mathbf{r})\, g(\mathbf{r})\; d\mathbf{r}$ for any function g using C samples of $g(\mathbf{r})$ as follows

$$\int_{\mathcal{R}} W(\mathbf{r})\, g(\mathbf{r})\; d\mathbf{r} \approx \sum_{c=1}^{C} w^c\, g(\mathbf{r}^c). \tag{5.1}$$

The sample locations $\mathbf{r}^c \in \mathcal{R}$ are known as the abscissae of the quadrature rule, and w^c are referred to as the weights. When the function $W(\mathbf{r})$ is the probability density function $p(\mathbf{r})$ of a continuous distribution, the quadrature rule may be used as an estimation of the expectation $\mathbb{E}_{p(\mathbf{r})}\left[g(\mathbf{r})\right]$. We construct a representation of the quadrature rule as a set of tuples of abscissae and weights, $\mathcal{Q}[p(\mathbf{r})] = \{(\mathbf{r}^c, w^c)\}_{c=1}^{C}$.

The abscissae and weights may be selected so that the approximation is exact when $g(\mathbf{r})$ is a polynomial up to a maximum order [80]. A quadrature rule that uses the fewest number of samples possible to exactly compute the expectation for polynomials of a given order is called a Gaussian quadrature rule. A C-point Gaussian quadrature rule in 1D can exactly integrate any polynomial of degree $2C-1$. Since large classes of functions can be approximated as polynomials, we make use of Gaussian quadrature rules to approximate expectations of broad classes of functions.

Gaussian quadrature rules for a large number of parameterized distributions are known. For example, Gaussian quadrature for the 1D function $W(r) = \exp(-r^2)$ is known as Gauss-Hermite quadrature. Scaling of Gauss-Hermite abscissae and weights may be used to approximate expectations over Gaussian distributions. When exact formulae for quadrature rules do not exist, a C-point Gaussian quadrature over any 1D distribution with finite moments may be computed using matrix operations on the first $2C-1$ moments of the distribution in $\mathcal{O}(C^3)$ operations [53]. For certain families of distributions, like Gaussian distributions, a Gaussian quadrature rule may be computed offline and parameterized in terms of the mean and variance of the distribution. In higher dimensions, non-Gaussian (and therefore suboptimal) quadrature rules may be computed by solving a linear program [40].

Quadrature rules are typically considered for continuous distributions, because computing expectations over discrete distributions can be performed exactly for every function g with a number of weighted samples equal to the size of the domain of the

discrete variable. In our discussion, we will use quadrature rules over distributions $p([\mathbf{r}, \mathbf{s}])$ with continuous component $\mathbf{r} \in \mathcal{R}$ and discrete component $\mathbf{s} \in \mathcal{S}$. These are constructed by applying a quadrature rule to $p(\mathbf{r} \mid \mathbf{s})$ for each $\mathbf{s}$, and multiplying the weights of those rules by $p(\mathbf{s})$. This yields

$$\mathcal{Q}[p(\mathbf{r}, \mathbf{s})] = \left\{([\mathbf{r}^c, \mathbf{s}^{c'}], w^c \times w^{c'}) \,|\, \mathbf{s}^{c'} \in \mathcal{S}, w^{c'} = p(\mathbf{s}^{c'}), (\mathbf{r}^c, w^c) \in \mathcal{Q}[p(\mathbf{r} \,|\, \mathbf{s}^{c'})]\right\}. \quad (5.2)$$

5.5 Preliminaries: Gaussian Processes

In this section, we introduce Gaussian processes and multi-output Gaussian processes formally. We will use this as the starting point for constructing the AcyGP model.

5.5.1 Single Attribute Gaussian Processes

A single-attribute Gaussian processes (GP) is a continuous valued regression model that models a function $y : \mathbb{R}^{D_x} \to \mathbb{R}$. Given a finite set of input/attribute pairs $\{(\boldsymbol{x}_i, y(\boldsymbol{x}_i))\}_{i=1}^N$, $\boldsymbol{x}_i \in \mathbb{R}^{D_x}$, where D_x is the dimensionality of $\boldsymbol{x}$, a GP models the variables $y(\boldsymbol{x}_i)$ as drawn from an N-dimensional Gaussian distribution [147]. Since the GP assigns a probability to any set of $(\boldsymbol{x}_i, y(\boldsymbol{x}_i))$ pairs, a GP places a prior distribution over the space of functions that model $y(\boldsymbol{x})$. Like any Bayesian prior, that prior can be updated in response to observed data to generate a posterior distribution. That posterior distribution gives a likelihood for the function conditioned on the data that has been observed.

The GP $f \sim \mathcal{GP}(m, k)$ is specified through its mean function $m(\cdot)$ and covariance kernel $k(\cdot, \cdot)$. The mean function $m(\boldsymbol{x}_i)$ controls the mean value of the process at any $\boldsymbol{x}_i$, and the kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ encodes the covariance between $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$. A single kernel is typically used for all $\boldsymbol{x}_i, \boldsymbol{x}_j$.

Denote the vectors of inputs and attributes as $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^T$ and $\mathbf{y} = [y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_N)]^T$. The attribute vector is modeled as the sum of values $\mathbf{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^T$ drawn from the Gaussian distribution. plus a vector $\boldsymbol{\epsilon}$ of

independent noise,

$$\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}, \tag{5.3}$$

$$\mathbf{f} \sim \mathcal{N}(m(\boldsymbol{X}), \boldsymbol{K}_{\mathbf{f},\mathbf{f}}), \tag{5.4}$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}), \tag{5.5}$$

where $[\boldsymbol{K}_{\mathbf{f},\mathbf{f}}]_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. We could have included noise in the kernel k, but it is separated out here to make explicit that $\mathbf{y}$ includes some noise on top of $\mathbf{f}$, and that we may be interested in estimating $\mathbf{f}$ without noise. This could be the case if $\boldsymbol{\epsilon}$ represents measurement noise, and we are interested in predicting a signal without that noise.

It is typical to shift the data such that $m(\boldsymbol{X}) = \mathbf{0}$ for training and prediction purposes, then perform the reverse transformation on predictions from the GP, for numerical stability. We follow this convention for the rest of this chapter, and focus only on GPs for which $m(\boldsymbol{X}) = \mathbf{0}$.

Prediction of attributes $\mathbf{y}^*$ at inputs $\boldsymbol{X}^*$ is accomplished using a conditional Gaussian distribution. The following equations follow from the mean and covariance of a Gaussian distribution with some variables observed [147].

$$\mathbf{f}^* \mid \mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{f}^*|\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{f}^*|\mathbf{y}}), \tag{5.6}$$

$$\boldsymbol{\mu}_{\mathbf{f}^*|\mathbf{y}} = \boldsymbol{K}_{\mathbf{f}^*,\mathbf{f}} \left[\boldsymbol{K}_{\mathbf{f},\mathbf{f}} + \sigma^2 \boldsymbol{I}\right]^{-1} \mathbf{y}, \tag{5.7}$$

$$\boldsymbol{\Sigma}_{\mathbf{f}^*|\mathbf{y}} = \boldsymbol{K}_{\mathbf{f}^*,\mathbf{f}^*} - \boldsymbol{K}_{\mathbf{f}^*,\mathbf{f}} \left[\boldsymbol{K}_{\mathbf{f},\mathbf{f}} + \sigma^2 \boldsymbol{I}\right]^{-1} \boldsymbol{K}_{\mathbf{f},\mathbf{f}^*}, \tag{5.8}$$

where $[\boldsymbol{K}_{\mathbf{f}^*,\mathbf{f}}]_{i,j} = k(\boldsymbol{x}_i^*, \boldsymbol{x}_j)$.

Gaussian process covariance kernels are selected so that the covariance matrix in the resultant Gaussian distribution is always positive definite for any choice of inputs $\boldsymbol{X}$. This requirement is necessary to ensure that any Gaussian distribution produced by the GP is well-defined, because Gaussian distributions are only defined for positive definite covariance matrices. The kernel parameterizes how the process at different

inputs is correlated, with larger numbers indicating greater correlation. To model correlation that decays with distance between inputs, many kernels decay to zero as a $|\boldsymbol{x}_i - \boldsymbol{x}_j|$ increases. Popular kernels include the radial basis function (RBF) kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta^2 \exp\left(-\frac{|\boldsymbol{x}_i - \boldsymbol{x}_j|^2}{2\, l_d^2}\right), \tag{5.9}$$

the rational quadratic kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta^2 \left(1 + \frac{|\boldsymbol{x}_i - \boldsymbol{x}_j|^2}{2\, \alpha\, l^2}\right)^{-\alpha}, \tag{5.10}$$

and the spectral mixture (SM) kernel [151]

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{s=1}^{S} \theta_s \prod_{d=1}^{D_x} \exp\left(-\frac{(x_{i,d} - x_{j,d})^2}{2\, l_{s,d}^2}\right) \cos\left(\nu_{s,d}\, (x_{i,d} - x_{j,d})\right). \tag{5.11}$$

The RBF kernel models smooth functions with a single length scale of correlations, the rational quadratic kernel is equivalent to a mixture of correlations over multiple length scales, and the SM kernel may be used to construct any pattern of correlation with enough elements.

Each kernel is controlled by a number of *hyperparameters*, including θ_s, $l_{s,d}$, $\nu_{s,d}$, and α in the kernels above, that are learned from the data. We denote the vector of parameters in the GP as $\boldsymbol{\theta}$. To train to the data, kernel hyperparameters and noise parameters are selected to maximize the log likelihood of the data, $\log p(\mathbf{y})$,

$$\log p_{\boldsymbol{\theta}}(\mathbf{y}) = -\frac{1}{2}\left(\mathbf{y}^T \left[\boldsymbol{K}_{\mathbf{f},\mathbf{f}} + \sigma^2 \boldsymbol{I}\right]^{-1} \mathbf{y} + \log\det\left[\boldsymbol{K}_{\mathbf{f},\mathbf{f}} + \sigma^2 \boldsymbol{I}\right] + n \log 2\pi\right). \tag{5.12}$$

which follows from likelihood of the gaussian distribution $\mathcal{N}(0, \boldsymbol{K}_{\mathbf{f},\mathbf{f}} + \sigma^2 \boldsymbol{I})$ used in the Gaussian process.

It is typical to perform this training using a quasi-Newton gradient descent algorithm, such as the limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS) [86]. L-BFGS performs quasi-Newton optimization, and constructs an approximation to the local inverse Hessian matrix from a small number of vectors, so

the full inverse Hessian is not held in memory as in BFGS [42].

5.5.2 Multi-Output Gaussian Processes

We now introduce multi-output Gaussian processes, and begin by introducing them in their general form. Multi-output Gaussian processes (MOGPs) extend GPs to model multiple attributes. We now consider the multiple attributes as a vector valued function in D_y dimensions, $\boldsymbol{y}(\boldsymbol{x}) = \left[y_1(\boldsymbol{x}), \ldots, y_{D_y}(\boldsymbol{x})\right]^T$. We define $\mathbf{f}_m$ and $\mathbf{y}_m$ as the Gaussian vector and attribute vector for the attribute with index m only, and define the vectors of all Gaussian values and attributes as $\mathbf{f}^T = \left[\mathbf{f}_1^T, \ldots, \mathbf{f}_{D_y}^T\right]$ and $\mathbf{y}^T = \left[\mathbf{y}_1^T, \ldots, \mathbf{y}_{D_y}^T\right]$ respectively. An MOGP expresses all attributes as a single ND_y-dimensional Gaussian distribution,

$$\mathbf{f} \sim \mathcal{N}(0, \boldsymbol{K}_{\mathbf{f},\mathbf{f}}), \quad \mathbf{f}^T = \left[\mathbf{f}_1^T \ \ldots \ \mathbf{f}_{D_y}^T\right]. \tag{5.13}$$

$\mathbf{K}_{\mathbf{f},\mathbf{f}}$ specifies all correlations between attributes, and may be considered to be composed of D_y^2 blocks of size $N \times N$,

$$\boldsymbol{K}_{\mathbf{f},\mathbf{f}} = \begin{bmatrix} \boldsymbol{K}_{\mathbf{f}_1,\mathbf{f}_1} & \cdots & \boldsymbol{K}_{\mathbf{f}_{D_y},\mathbf{f}_1} \\ \vdots & \ddots & \vdots \\ \boldsymbol{K}_{\mathbf{f}_{D_y},\mathbf{f}_1} & \cdots & \boldsymbol{K}_{\mathbf{f}_{D_y},\mathbf{f}_{D_y}} \end{bmatrix}. \tag{5.14}$$

The block $\boldsymbol{K}_{\mathbf{f}_m,\mathbf{f}_n}$ corresponding to attributes m and n is modeled using a cross covariance kernel specific to attributes m and n, so that

$$[\boldsymbol{K}_{\mathbf{f}_m,\mathbf{f}_n}]_{i,j} = k_{mn}(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{5.15}$$

The kernels are defined such that the overall covariance matrix is positive definite.

5.5.3 Fully Correlated MOGPs from Latent Processes

The MOGPs we introduced in the previous section place few restrictions on the cross covariance kernels except that the overall covariance matrix must be positive definite,

but it can be difficult to formulate and optimize with this constraint in practice. One way to construct an MOGP with a covariance that is necessarily positive definite is to construct it as a linear combination of unobserved (latent) single-attribute GPs. We now discuss this construction, which will be useful as a starting point to develop the AcyGP model when all data is continuous.

Linearly combining multiple independent single-attribute latent GPs introduces correlations between the attribute dimensions. The semiparametric latent factor model (SLFM) [132] models each attribute as the weighted linear combination of Q independent latent processes u_q, so that

$$f_m(\boldsymbol{x}) = \sum_{q=1}^{Q} \boldsymbol{\Lambda}_{m,q}\ u_q(\boldsymbol{x}), \tag{5.16}$$

where each $u_q \sim \mathcal{GP}(0, \tilde{k}_q)$ is an independent single-attribute GP and $\boldsymbol{\Lambda}$ is a $D_y \times Q$ matrix of scalar coefficients. The SLFM model is visualized in Figure 5-4a. The SLFM model induces the following effective covariance kernel between attributes

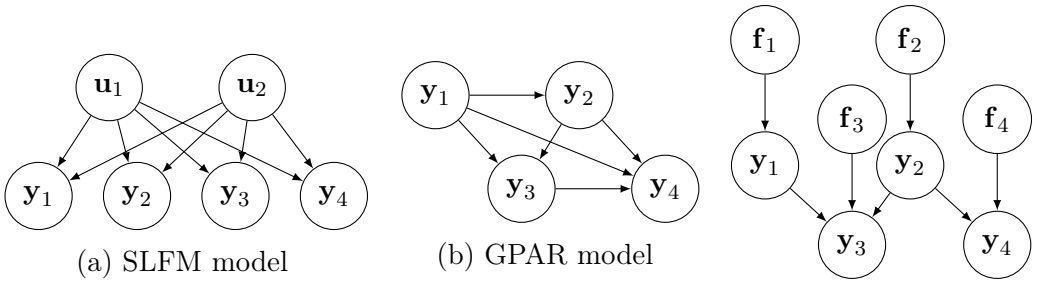
$$k_{mn}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{q=1}^{Q} \boldsymbol{\Lambda}_{m,q}\ \boldsymbol{\Lambda}_{n,q}\ \tilde{k}_q(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{5.17}$$

For example, when using RBF kernels for all latent processes, each kernel k_{mn} encodes correlations over Q length scales in the SLFM.

Then, as in the single attribute case, dimension m of the attributes is constructed from the Gaussian signal and independent noise, following $\mathbf{y}_m = \mathbf{f}_m + \boldsymbol{\epsilon}_m$, where $\boldsymbol{\epsilon}_m \sim \mathcal{N}(0, \sigma_m^2 \boldsymbol{I})$.

5.5.4 MOGP Networks

Alternative methods seek to control the dependence between attributes by explicitly constructing a network structure between attributes. The Gaussian processive autoregressive regression model (GPAR) [110] constructs each attribute using a GP



(a) SLFM model (b) GPAR model (c) AcyGP model

Figure 5-4: Comparison of MOGP model structures.

trained on vectors with previous attributes,

$$\begin{aligned} y_1(\boldsymbol{x}) &= f_1(\boldsymbol{x}) + \epsilon_1 \\ y_2(\boldsymbol{x}) &= f_2([\boldsymbol{x}, y_1(\boldsymbol{x})]) + \epsilon_2 \\ y_3(\boldsymbol{x}) &= f_3([\boldsymbol{x}, y_1(\boldsymbol{x}), y_2(\boldsymbol{x})]) + \epsilon_3 \\ &\vdots \end{aligned} \tag{5.18}$$

A schematic of GPAR is shown in Figure 5-4b. In principle, the GP f_m can be constructed to depend on only a subset of previous attributes [1]. However, the ordering and choice of how gaussian processes are related has been hand-coded in previous applications of this technique. For many domains, it is not immediately clear how to construct the network of dependencies, or only partial knowledge of the correct correlations may be known.

5.6 Preliminaries: Directed Acyclic Graphical Models

A directed acyclic graphical model (DAG) $\mathcal{G} = (\mathcal{Y}, \mathcal{E})$ consists of a set of variables $\mathcal{Y}$ and directed edges $\mathcal{E}$. There is no requirement that each variable in $\mathcal{Y}$ is scalar, and for our purposes we will consider $\mathcal{Y}$ to be composed of random vectors $\mathbf{y}_m$, with $m \in [D_y]$, where $[D_y] = \{1, 2, \ldots, D_y\}$. Each directed edge in $\mathcal{E}$ points from and to elements of $\mathcal{Y}$, written as $\mathbf{y}_m \to \mathbf{y}_n$. Acyclicity in the DAG asserts that it is not

possible to create a loop by following directed edges in their indicated direction.

In a DAG model, a variable is independent of its non-descendants given its parents. This implies that a distribution $p_{\boldsymbol{\theta}}(\mathbf{y})$ that factorizes over a DAG $\mathcal{G}$ be written as

$$p_{\boldsymbol{\theta}}(\mathbf{y}) = \prod_{m=1}^{D_y} p_{\boldsymbol{\theta}_m}(\mathbf{y}_m \,|\, \mathbf{y}_{\Pi_m^{\mathcal{G}}}), \tag{5.19}$$

where $\boldsymbol{\theta}_m$ is the vector of parameters associated with the distribution of $\mathbf{y}_m \mid \mathbf{y}_{\Pi_m^{\mathcal{G}}}$, and $\Pi_m^{\mathcal{G}}$ refers to the *parent set* of $\mathbf{y}_m$, defined as the indices of variables with directed edges pointing to $\mathbf{y}_m$, i.e. $\Pi_m^{\mathcal{G}} = \{n \,|\, (\mathbf{y}_n \to \mathbf{y}_m) \in \mathcal{E}\}$, and $\mathbf{y}_{\Pi_m^{\mathcal{G}}} = \{\mathbf{y}_n \,|\, n \in \Pi_m^{\mathcal{G}}\}$. This factorization of the probability distribution implies a direct dependence of a variable on its parents, and so the DAG structure provides a means of controlling conditional independence.

Directed acyclic graphs are one of several methods of relating conditional independence between variables to structure. Alternate methods include undirected graphical models and hybrid graphs [43]. We are particularly interested in DAGs for developing the AcyGP model because a DAG can be easily constructed through a *structural equation model*, as described below. A structural equation model describes a set of variables $\mathcal{Y}$ using a set of stochastic functions $\{F_m\}$ according to

$$\mathbf{y}_m = F_m(\mathbf{y}_{\Pi_m^{\mathcal{G}}}). \tag{5.20}$$

If this model is satisfied, then $p_{\boldsymbol{\theta}}(\mathbf{y})$ necessarily factorizes according to the DAG $\mathcal{G}$. This follows from the fact that F_m implies that $\mathbf{y}_m$ depends only on $\mathbf{y}_{\Pi_m^{\mathcal{G}}}$, leading to a factorization of the form in (5.19).

For example, consider a Gaussian graphical model where $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3\}$ are jointly Gaussian distributed, and $\mathcal{E} = \{(\mathbf{y}_1 \to \mathbf{y}_3), (\mathbf{y}_2 \to \mathbf{y}_3)\}$. Then we can describe

the variables through the following structural equation model,

$$\begin{aligned}
\mathbf{y}_1 &\sim \mathcal{N}(\mu_1, \sigma_1^2) \\
\mathbf{y}_2 &\sim \mathcal{N}(\mu_2, \sigma_2^2) \\
\mathbf{y}_3 &= a_1\mathbf{y}_1 + a_2\mathbf{y}_2 + \mathbf{z}_3, \quad \mathbf{z}_3 \sim \mathcal{N}(\mu_3, \sigma_3^2).
\end{aligned}$$

5.6.1 Topological Orderings

Topological orderings can be used to draw samples from a graph $\mathcal{G}$ by sampling each variable in the order given by the topological ordering. We will use samples to perform training and prediction for the AcyGP model, so in this section we introduce how those topological orderings may be found.

Since a DAG permits no cycles in its edges, an ordering can be defined over its variables so that parents always appear earlier than children. A *topological ordering* of graph $\mathcal{G} = (\mathcal{Y}, \mathcal{E})$ is an ordered list $T(\mathcal{G}) = (\mathbf{y}_{m_1}, \mathbf{y}_{m_2}, \ldots, \mathbf{y}_{m_{D_y}})$ such that for all $(\mathbf{y}_n \to \mathbf{y}_m) \in \mathcal{E}$, $\mathbf{y}_n$ comes before $\mathbf{y}_m$ in $T(\mathcal{G})$. A single DAG may have multiple topological orderings. A sample can then be drawn from $\mathcal{G}$ by sampling each variable $\mathbf{y}_m$ from the distribution $p_{\boldsymbol{\theta}_m}(\mathbf{y}_m \,|\, \mathbf{y}_{\Pi_m^{\mathcal{G}}})$ in order of $T(\mathcal{G})$.

A topological ordering is constructed by repeatedly extending a topological ordering over a subset of the variables, until all variables are ordered. We will use the notation $T_{m_k}(\mathcal{G}) = (\mathbf{y}_{m_1}, \mathbf{y}_{m_2}, \ldots, \mathbf{y}_{m_k})$ to refer to a topological ordering up to $\mathbf{y}_{m_k}$. Since topological orderings are not unique, the notation $T_{m_k}(\mathcal{G})$ may be ambiguous. In our algorithms that construct $T(\mathcal{G})$ incrementally, $T_{m_k}(\mathcal{G})$ will refer to a partial topological ordering constructed at a previous step.

A topological ordering may be constructed for $\mathcal{G}$ by starting with the empty list $T_\emptyset(\mathcal{G}) = ()$, and incrementally constructing $T_{m_{k+1}}(\mathcal{G}) = (T_{m_k}(\mathcal{G}), \mathbf{y}_{m_{k+1}})$ when $\mathbf{y}_{m_{k+1}}$ has all parents already in the list $T_{m_k}(\mathcal{G})$.

5.6.2 Score Based Structure Learning

When training an AcyGP, it will frequently be the case that the full DAG structure of is unknown, but observations drawn from the environment model are available. We will use structure learning to infer the unknown parts of the structure from the observational data. In this section, we review the principles of score based structure learning for directed acyclic graphical models.

In our context, structure learning refers to inferring the DAG structure of a distribution that was used to produce data samples, in addition to any parameters associated with that DAG. A partial structure may be known and can be accommodated as a constraint, but this is not necessary.

Performing structure learning has two main advantages over assuming a fully connected network. First, the structure itself may be explanatory about the distribution, by revealing causal relationships or independence between variables that were not obvious from the data alone. Second, learning a structure and then performing prediction using that model has been repeatedly empirically demonstrated to improve prediction accuracy [43]. When a distribution factorizes according to a DAG, it typically may be described using fewer parameters compared to an arbitrary distribution. For example, learning a Gaussian DAG requires learning means and variances for each variable, plus edge weights. When the DAG is sparse, this is fewer parameters than learning means and a joint covariance matrix for all variables. Performing inference without introducing all the unnecessary parameters in a fully correlated distribution allows parameters to be learned from data more accurately.

Furthermore, statistical noise in the data can be recognized by its deviation from the model, making parameter learning more robust to noise. For example, data drawn from a Gaussian graphical model with additional noise may be well modeled by an arbitrary multivariate Gaussian distribution with a certain covariance matrix, but may not be well modeled with any Gaussian distribution that factorizes according to the DAG. In this case, finding the closest distribution that factorizes according to the DAG gives a more accurate model, rather than fitting to the noise.

It is expected that the observed data $\mathbf{y}$ will exhibit high likelihood under an accurate model. However, for DAGs representing most distributions of interest, the maximum possible likelihood grows with DAG complexity, and the maximum likelihood model will always be a fully connected DAG, regardless of the true underlying model. This makes selecting the maximum likelihod model an unsuitable objective for structure learning. Concretely, for graphs $\mathcal{G} = (\mathcal{Y}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{Y}, \mathcal{E}')$ with $\mathcal{E} \subseteq \mathcal{E}'$, and distributions parameterized by vectors $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, we typically have

$$\max_{\boldsymbol{\theta}} \; p_{\boldsymbol{\theta}}(\mathbf{y} \,|\, \mathcal{G}) \leq \max_{\boldsymbol{\theta}'} \; p_{\boldsymbol{\theta}'}(\mathbf{y} \,|\, \mathcal{G}'). \tag{5.21}$$

Score based structure learning methods instead maximize a *score* that increases with increasing data likelihood, and decreases with increased DAG complexity. The idea is to find a simple DAG that matches the data well, which will give better predictions of future samples. However, unlike maximum parsimony [138], which uses domain specific measures of complexity to construct a simple tree from the data, score based methods are usually based on probabilistic likelihood and a count of the number of edges in a graph.

For DAG structure learning, the score is typically the accumulation of scores for individual variables, where the individual variable scores depend only on their parents, so that the graph selected satisfies

$$\arg\max_{\mathcal{G}} \; \sum_{m=1}^{D_y} Score(\mathbf{y}_m, \mathbf{y}_{\Pi_m^{\mathcal{G}}}). \tag{5.22}$$

When using such a score, scores for each variable and parent set may be evaluated separately, which reduces the total number of scores that need to be computed compared to a unique score for every possible DAG. Optimization consists of selecting the structure that optimizes the sum of scores while maintaining acyclicity in the graph.

Scores depend on data likelihood, which is a function of unknown parameters $\boldsymbol{\theta}$. We may place a prior over $\boldsymbol{\theta}$ to compute data likelihood, but it may be difficult to define accurate prior distributions $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathcal{G})$, particularly before any data has

been observed. For these scenarios, scores that are independent of $p(\boldsymbol{\theta})$ have been developed, including the Akaike information criterion (AIC) [2] and the Bayesian information criterion (BIC) [119] were developed. AIC and BIC are both designed to be minimized, so their negatives give scores that must be maximized. For a given graphical model $\mathcal{G}$ with distributions parameterized by $\boldsymbol{\theta}$ and N data samples, AIC and BIC scores are defined as

$$\text{AIC} = \sum_{m=1}^{D_y} 2|\boldsymbol{\theta}_m| - 2 \max_{\boldsymbol{\theta}_m} \log p_{\boldsymbol{\theta}_m}(\mathbf{y}_m \,|\, \mathbf{y}_{\Pi_m^{\mathcal{G}}}) \tag{5.23}$$

$$\text{BIC} = \sum_{m=1}^{D_y} |\boldsymbol{\theta}_m| \log N - 2 \max_{\boldsymbol{\theta}_m} \log p_{\boldsymbol{\theta}_m}(\mathbf{y}_m \,|\, \mathbf{y}_{\Pi_m^{\mathcal{G}}}). \tag{5.24}$$

When the data is actually generated by a model in the model candidate set, BIC is known to be consistent, meaning that minimizing BIC recovers the true structure with probability approaching 1 in the large data limit. This property holds even with correlated data [26]. By contrast, AIC is derived under assumptions that all models are approximations to a more complex data-generating process. In this thesis, we will primarily use BIC, following an assumption that the AcyGP model is accurate for a certain choice of structure and parameters.

5.7 Intuition Behind the AcyGP Model

The AcyGP model performed multi-output Gaussian process regression by combining single-attribute Gaussian processes in a DAG structure. We claim that the AcyGP model is essential to avoiding spurious correlations that arise, given the limited data that is available during adaptive sampling. In this section, we discuss the intuition behind this idea, including why a DAG structure is expected to improve prediction accuracy in an MOGP.

To understand why spurious correlations between attributes are introduced, consider applying MOGP inference to two attributes, $y_1(\boldsymbol{x})$ and $y_2(\boldsymbol{x})$. Unbeknown to the modeler, the two functions are independent. As an example, both y_1 and y_2

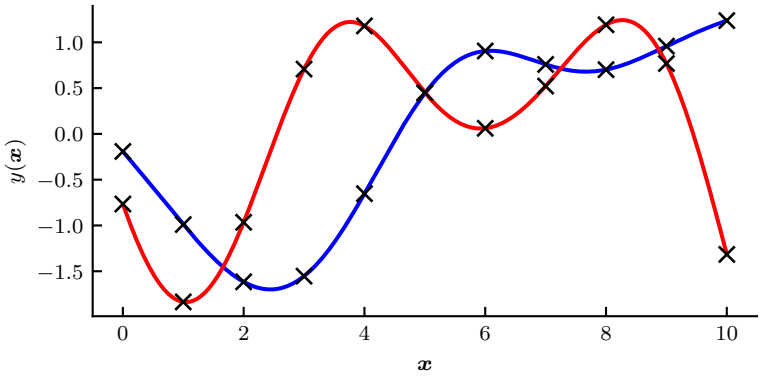


Figure 5-5: Samples fro two independent GPs with positive correlation.

may represent time series data of humidity from different weather stations, but the stations are separated by such a large distance that the weather patterns are effectively unrelated. Alternatively, y_1 may represent the methane concentrations at each location $\boldsymbol{x}$ in the ocean, while y_2 represents effective density of the sea water.

Despite the independence of y_1 and y_2, non-zero correlations between the attributes are likely to be observed in the presence of limited data; we illustrate as follows. Figure 5-5 shows 11 samples from two independently modeled Gaussian processes y_1 and y_2. In both cases, the data is normalized to mean 0 and standard deviation 1. By chance, both tend to increase at similar points in time, which allows computation of a positive covariance, in this case of 0.205.

A sufficiently large set of data drawn from the GPs would show that y_2 decreases as often as it increases when y_1 increases, but with small data sets like we encounter in adaptive sampling, statistically arising correlations are likely. In fact, when randomly generating normalized points from two independent Gaussian processes in the way described above, we experimentally observed that $|cov(y_1, y_2)| > 0.2$ approximately 80% of the time.

Existing MOGP methods will identify and learn these correlations, and erroneously use the behavior of y_1 to influence prediction of y_2. More accurate results would instead be obtained by training each attribute dimension as an independent GP. The AcyGP model assesses the strength of the observed correlations relative to the size of the data set, determines that independence between y_1 and y_2 is likely, and

converges on a model with y_1 independent of y_2. Use of the AcyGP model results in lower error in model predictions, and frequently allows for meaningful interpretations of the dependencies between variables to be derived from the data.

5.8 The Homogeneous AcyGP Model

We now describe our AcyGP model, including how it is structured, how parameters and structure are learned, and how to perform prediction with the trained model. It is conceptually useful to begin with a description of the simpler *homogeneous model*, where all attributes are assumed to be continuous, real valued, and unbounded. This excludes discrete valued attributes, or an attribute bounded between 0 and 1, for example. This is the standard modeling assumption for a Gaussian process, and it permits likelihood computation and prediction to be performed in closed form. However, we acknowledge that the assumption of continuous unbounded real valued data is inappropriate for many adaptive sampling missions, so extensions to more complex attributes are considered in Section 5.10.

5.8.1 Homogeneous AcyGP Construction

An AcyGP enforces a DAG $\mathcal{G}$ between the D_y attributes. $\mathcal{G}$ is described by parent sets $\Pi_m^{\mathcal{G}}$ for each attribute, containing the indices of parents of attribute m. Each attribute is then modeled as a combination of its parent attributes and a latent single attribute Gaussian process.

In analogy to the structural equation model of a Gaussian DAG, a heterogeneous AcyGP combines latent GPs $f_m \sim \mathcal{GP}(0, k_m)$, white noise processes ε_m, and a weighted adjacency matrix $\mathbf{\Lambda} \in \mathbb{R}^{D_y \times D_y}$ according to

$$y_m(\boldsymbol{x}) = f_m(\boldsymbol{x}) + \varepsilon_m(\boldsymbol{x}) + \sum_{n \in \Pi_m^{\mathcal{G}}} \mathbf{\Lambda}_{m,n}\, y_n(\boldsymbol{x}). \tag{5.25}$$

Element m, n of the adjacency matrix is the coefficient that parent attribute y_n should be multiplied by when constructing y_m. For all $n \notin \Pi_m^{\mathcal{G}}$, it holds that $\mathbf{\Lambda}_{m,n} = 0$.

Kernels k_m are selected by the modeler when an AcyGP is constructed.

This construction is a structural equation model between the attributes, so the attributes factor according to the DAG $\mathcal{G}$, so that for any vector of inputs $\boldsymbol{X}$, the vector of each attribute at $\boldsymbol{X}$ is independent of its non-descendents in the graph at $\boldsymbol{X}$, given values for its parents at $\boldsymbol{X}$. Formally, $\boldsymbol{y}_m(\boldsymbol{X}) \perp \boldsymbol{y}_{ND_m^{\mathcal{G}}}(\boldsymbol{X}) \,|\, \mathbf{y}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{X})$, where $ND_m^{\mathcal{G}}$ is the set of indices of non-descendants of attribute m.

The selected DAG structure therefore provides a mechanism to control the dependencies between observables in an environment, and the structure may be interpreted as a declaration of a causal model between attributes. Introducing conditional independence in the model in this manner ensures correlations between independent attributes are also independent, improving prediction accuracy.

We can write a simple expression connecting the vector functions $\boldsymbol{y}(\boldsymbol{x})$, $\boldsymbol{f}(\boldsymbol{x})$, and $\boldsymbol{\varepsilon}(\boldsymbol{x})$. First define the matrix $\boldsymbol{B} = \boldsymbol{I} - \boldsymbol{\Lambda}$. Then

$$\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\varepsilon}(\boldsymbol{x}) = \boldsymbol{B}\,\boldsymbol{y}(\boldsymbol{x}). \tag{5.26}$$

Recognition of this connection between $\boldsymbol{y}$, $\boldsymbol{f}$, and $\boldsymbol{\varepsilon}$ will be key for training and performing inference in the AcyGP.

5.8.2 Comparison to Existing MOGP Methods

The model described in (5.25) expresses each attribute as a local linear transformations of parent attributes. The transformation is local because $y_m(\boldsymbol{x})$ depends on y_n only at $\boldsymbol{x}$, and it is linear because y_n is multiplied by a coefficient. Local linear transformations are known to be effective in MOGP models [83], and they permit inference to be performed exactly and in closed form because the resulting model is fully Gaussian. The homogeneous AcyGP model bears similarity to SLFM in (5.16) in that each attribute is constructed from a linear combination of latent and noise processes. However, unlike SLFM, not all attributes are correlated in the AcyGP model, so spurious correlations inconsistent with structure are not learned, and the noise processes from ancestor attributes are included in each attribute. The formula-

tion also bears similarity to MOGP networks when the function descriptions in (5.18) are formulated to have linear dependence on previous attributes. Unlike GPAR, not all previous attributes are used as inputs to the next attribute. Compared to these fully connected models, fewer edges prevents spurious correlations being introduced between attributes, and improves prediction accuracy, as discussed in Section 5.6.

The AcyGP model also differs from alternative MOGOP networks that allow flexibility in structure [1, 70, 83], but require the modeler to hand-code the dependencies between attributes. In an AcyGP, the parent sets may model any DAG, and unknown edges are learned from the data, which allows conditional independencies between attributes to be enforced.

5.8.3 Selection of Parameters and DAG Structure

When constructing an AcyGP model, we allow an expert to define relationships between attributes that are known to exist or not to exist. The remainder of the DAG is then chosen so that it provides the best fit to the data, consistent with the relationships defined by the expert. The expert supplies the known relationships in terms of allowable parent sets for each index, denoted by $\{\Xi_m\}_{m=1}^{D_y}$. Each Ξ_m contains all permissible sets of parents for attribute y_m. For any DAG $\mathcal{G}$ that could be found to be the structure for the AcyGP, $\Pi_m^{\mathcal{G}} \in \Xi_m$. Our structure learning algorithm does not require this set to be explicitly enumerated. The expert may instead supply a function that returns whether $\Pi_m^{\mathcal{G}} \in \Xi_m$ holds. This way, the set Ξ_m may be defined through rules, for example that a specific parent may not exist, or that the total number of parents must be less than a certain size.

When we train an AcyGP, we search for the vector $\boldsymbol{\theta}$ of kernel parameters and parent weights and the DAG $\mathcal{G}$ consistent with $\{\Xi_m\}$ that minimizes the Bayesian Information Criterion [119]. BIC is selected because it remains consistent when data at different inputs is correlated. BIC maximizes likelihood of all data $\mathbf{y}$, and penalizes edges in $\mathcal{G}$, ensuring that edges are only included if they lead to a sufficient increase in likelihood.

When training an AcyGP, we do not change the choices of covariance kernels k_m.

This means that the number of kernel hyperparameters stays the same regardless of DAG structure. For example, an AcyGP constructed with RBF kernels always has a variance scale parameter and a length scale parameter. In contrast, changing the number of edges in the DAG will change the number of parameters in the AcyGP. Non-zero entries of the matrix $\mathbf{\Lambda}$ may be considered to be parameters, and there is one for every edge. If $\mathbf{y}_n \to \mathbf{y}_m$ is not in the selected structure, then $\mathbf{\Lambda}_{m,n}$ is 0 by construction, and is not considered a variable parameter of the model.

Since the number of kernel parameters does not change, we may exclude them from the BIC computation. In the homogeneous AcyGP model, a single parameter $\mathbf{\Lambda}_{m,n}$ is specified for each parent $\mathbf{y}_n$ of variable $\mathbf{y}_m$, so we maximize the following expression equivalent to minimizing BIC,

$$\sum_{m=1}^{D_y} \max_{\boldsymbol{\theta}_m} \log p_{\boldsymbol{\theta}_m}(\mathbf{y}_m \,|\, \mathbf{y}_{\Pi_m^{\mathcal{G}}}) - \frac{|\Pi_m^{\mathcal{G}}|}{2} \log N \quad \text{such that } \Pi_m^{\mathcal{G}} \in \Xi_m \tag{5.27}$$

Training a homogeneous AcyGP is therefore performed by searching for the parameters $\boldsymbol{\theta}$ and the graphical model $\mathcal{G}$ that satisfies (5.27). This objective may be viewed as maximizing log likelihood, with a term that penalizes additional edges in the DAG $\mathcal{G}$. We express this in the following problem, and describe how this is performed in the following section.

Problem 3. *Training a homogeneous AcyGP.*

Given observations $\{\mathbf{y}_m\}_{m=1}^{D_y}$ *at* N *inputs, kernels* $\{k_m\}_{m=1}^{D_y}$*, and sets of candidate parent sets* $\{\Xi_m\}_{m=1}^{D_y}$*, determine the DAG structure* $\mathcal{G}^*$ *and vector of parameters* $\boldsymbol{\theta}^*$*, consisting of kernel hyperparameters and AcyGP edge parameters* $\mathbf{\Lambda}_{m,n}$*, that satisfy*

$$\boldsymbol{\theta}^*, \mathcal{G}^* = \arg\max_{\boldsymbol{\theta}_m, \mathcal{G}} \sum_{m=1}^{D_y} \log p_{\boldsymbol{\theta}_m}(\mathbf{y}_m \,|\, \mathbf{y}_{\Pi_m^{\mathcal{G}}}) - \frac{|\Pi_m^{\mathcal{G}}|}{2} \log N$$

under a homogeneous AcyGP model, such that $\Pi_m^{\mathcal{G}} \in \Xi_m \ \forall m \in [D_y]$.

5.8.4 Optimization Procedure for Parameters and DAG Structure

We now discuss how to solve for the parameters and structure that solve Problem 3. When all attributes are observed, the parameters for a known structure can be trained by optimizing D_y single attribute GPs, as in (5.25), each depending only on $\mathbf{y}_m$ and $\mathbf{y}_{\Pi_m^{\mathcal{G}}}$. Factorization in this manner is critical for structure learning, because it allows reasoning over combinations of separate parent sets, instead of optimizing the parameters for every possible DAG individually. However, when parents are unobserved, marginalization will cause an attribute to depend on further ancestors.

As an example, consider an AcyGP where y_2 has y_1 as its only parent, and y_3 has y_2 as its only parent. If $y_1(\boldsymbol{x})$, $y_2(\boldsymbol{x})$, and $y_3(\boldsymbol{x})$ have all been observed, then the parameters of the processes f_2 and ε_2 can be selected to maximize the likelihood of $y_2(\boldsymbol{x}) - \boldsymbol{\Lambda}_{2,1} y_1(\boldsymbol{x})$ as a single attribute Gaussian process, and the same procedure can be used to optimize f_3 and ε_3. But if we have only observed $y_1(\boldsymbol{x})$ and $y_3(\boldsymbol{x})$, then since $y_2(\boldsymbol{x})$ is unobserved, $y_3(\boldsymbol{x})$ depends directly $y_1(\boldsymbol{x})$. In effect, the distribution no longer factors according to its DAG structure when data is missing, so we cannot compute BIC as a separable sum for each variable, greatly complicating structural optimization.

In the cases when some attributes are missing, $\boldsymbol{\theta}$ and $\mathcal{G}$ are selected using the Structural EM algorithm [44]. Structural EM simultaneously optimizes parameters and structure that maximize a penalized likelihood, as desired in Problem 3, in the presence of limited data.

The basic idea of (non-structural) expectation maximization (EM) is determine parameters $\boldsymbol{\theta}$ that maximize the log likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{y})$ of some observed data $\mathbf{y}$, while there also exists some data $\mathbf{w}$ that is unobserved. Maximization of likelihood of the observed data alone is intractable, but optimizing the log likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{w})$ is simpler. EM determines the optimal parameters in an iterative procedure. At iteration 0 it assumes an initial value $\boldsymbol{\theta}^{(0)}$ for the parameters. At iteration s, given $\boldsymbol{\theta}^{(s)}$, it selects $\boldsymbol{\theta}^{(s+1)}$ that optimizes $\mathbb{E}_{\mathbf{w}|\mathbf{y},\boldsymbol{\theta}^{(s)}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{w})\right]$. This repeats until convergence

of $\boldsymbol{\theta}$, at which point the parameters are a local maximum of $p_{\boldsymbol{\theta}}(\mathbf{y})$.

Structural EM works very similarly. There is observed data $\mathbf{y}$, and unobserved data $\mathbf{w}$ that depend upon parameters $\boldsymbol{\theta}$ and a structural model $\mathcal{G}$. The objective is to maximize $\log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathcal{G}) - c(\mathcal{G})$ for a function c that depends only on the structure. In the AcyGP case, $\mathbf{y}$ is observed data, $\mathbf{w}$ is unobserved attributes at the inputs where $\mathbf{y}$ is observed, and the form of c is given by (5.27). If there was no unobserved data, the likelihood would factor between parent sets for any DAG model. This would mean that $\boldsymbol{\theta}_m$ could be found that optimized $\log p_{\boldsymbol{\theta}_m}(\mathbf{y}_m \mid \mathbf{y}_{\Pi_m^{\mathcal{G}}})$ for each $\Pi_m^{\mathcal{G}}$, and then structure could be optimized as a combinatorial search over parent sets. But since certain data is missing, the likelihood does not factor, and optimizing the parameters and structure is more difficult. Like in EM, structural EM assumes values of $\boldsymbol{\theta}$ and $\mathcal{G}$ and updates them iteratively to values that optimize an expected log likelihood.

A description of Structural EM for the AcyGP model is given in Algorithm 15. We first construct the vector of estimated values of observed and unobserved data $\mathbf{v}^T = [\mathbf{v}_1^T, \ldots, \mathbf{v}_{D_y}^T]$. Each $\mathbf{v}_m$ contains the observed data $\mathbf{y}_m$ in addition to $\mathbf{w}_m$, which consists of estimations of the unobserved values out of $y_m(\boldsymbol{x}_1), \ldots, y_m(\boldsymbol{x}_N)$. For iteration counters s and t, structural EM iteratively optimizes the expectation

$$\mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}}\left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v} \mid \mathcal{G})\right] - \sum_{m=1}^{D_y} \frac{|\Pi_m^{\mathcal{G}}|}{2} \log N, \tag{5.28}$$

and selects the next iterations of $\boldsymbol{\theta}$ and $\mathcal{G}$ to be the parameters and structure that optimized this objective. Since $\mathbf{v}$ contains all attributes, we know that $\log p_{\boldsymbol{\theta}_m}(\mathbf{v} \mid \mathcal{G}) = \sum_{m=1}^{D_y} \log p_{\boldsymbol{\theta}_m}(\mathbf{v} \mid \mathbf{v}_{\Pi_m^{\mathcal{G}}})$, so the objective to optimize factors as

$$\sum_{m=1}^{D_y} \mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}}\left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \mid \mathbf{v}_{\Pi_m^{\mathcal{G}}})\right] - \frac{|\Pi_m^{\mathcal{G}}|}{2} \log N. \tag{5.29}$$

At convergence, the solution is a local optimum of BIC.

Optimization in Algorithm 15 is done in an inner loop and an outer loop. In the inner loop on line 2, only the parameters $\boldsymbol{\theta}$ are optimized while the structure is held constant. This loop is run for a set number of iterations s_{max} or until convergence.

Algorithm 15: Structural EM Algorithm

Input : Initial parameters $\boldsymbol{\theta}^{(0,0)}$ and DAG $\mathcal{G}^{(0)}$
Output: Maximum likelihood hyperparameters $\boldsymbol{\theta}^{(t,s)}$ and DAG $\mathcal{G}^{(t)}$, possible parent sets $\{\Xi_m\}$

1 **loop** for $t = 0, 1, \ldots$ until convergence
2 **loop** for $s = 0, 1, \ldots$ until $s = s_{max}$ or convergence
3 $\boldsymbol{\theta}^{(t,s+1)} \leftarrow \arg\max_{\boldsymbol{\theta}} \sum_{m=1}^{D_y} \mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}} \left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}})\right]$
4 $\boldsymbol{\theta}^{(t+1,0)}$, $\mathcal{G}^{(t+1)} \leftarrow$ $\arg\max_{\boldsymbol{\theta},\mathcal{G}} \sum_{m=1}^{D_y} \mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}} \left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}})\right] - \frac{|\Pi_m^{\mathcal{G}}|}{2} \log N$ such that $\Pi_m^{\mathcal{G}} \in \Xi_m$

Since the structure is held constant, structure penalization can be excluded from the objective in the inner loop. The outer loop on line 1 runs the inner loop, and once convergence is reached, then performs simultaneous optimization of $\boldsymbol{\theta}$ and the structure $\mathcal{G}$ on line 4. Optimization of structure is performed less frequently because it is typically more computationally demanding than optimizing parameters for a fixed structure, so optimizing parameters more frequently accelerates the algorithm.

Line 4 of Structural EM is achieved through the use of score-based structure learning. We optimize $\boldsymbol{\theta}_m$ for candidate parent sets in order to assign them the following score,

$$\sum_{m=1}^{D_y} Score^{(t,s)}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) = \sum_{m=1}^{D_y} \hat{\mathcal{L}}^{(t,s)}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) - \frac{|\Pi_m^{\mathcal{G}}|}{2} \log N \tag{5.30}$$

$$\hat{\mathcal{L}}^{(t,s)}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) := \max_{\boldsymbol{\theta}_m} \; \mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}} \left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}})\right]. \tag{5.31}$$

Each DAG is assigned a score equal to the sum of the scores of its parent sets. The chosen structure $\mathcal{G}^{(t+1)}$ is found by evaluating the scores of parent sets that appear in $\{\Xi_m\}$, and searching for the combination with highest combined score that does not introduce cycles. This may be done naively by evaluating the score for all possible parent sets, but this is highly inefficient. Discussion of how to perform this search efficiently is the focus of the following chapter.

The primary difference between our use of the structural EM algorithm and its original proposal by Friedman [44] is that our data is correlated between inputs be-

cause it is drawn from a Gaussian process, rather than independent draws from a graphical model. This means that the $\mathbf{v}_m$ we consider are correlated across inputs, with optimization of $\boldsymbol{\theta}$ requiring a numerical optimization procedure.

5.8.5 Optimization Procedure for a Candidate Structure

A key step in the Structural EM algorithm is to find parameters $\boldsymbol{\theta}_m$ that optimize $\mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}}\left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}})\right]$. This is done on line 3 of Algorithm 15, and for all possible parent sets as part of the optimization on line 4. We discuss its computation in this section.

Since all variables in the homogeneous AcyGP model are Gaussian distributed, expected log likelihood for an attribute with a candidate parent set can be computed in closed form. Let $\mathbf{v} \,|\, \mathbf{y}, \boldsymbol{\theta}^{(t,s)}, \mathcal{G}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{v}|\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{v}|\mathbf{y}})$, and construct the $N \times D_y N$ matrix $\boldsymbol{A}_m$ as

$$\boldsymbol{A}_m = \left[\begin{array}{cccc} \boldsymbol{B}_{m,1}\boldsymbol{I} & \boldsymbol{B}_{m,2}\boldsymbol{I} & \cdots & \boldsymbol{B}_{m,D_y}\boldsymbol{I} \end{array} \right], \tag{5.32}$$

where $\boldsymbol{B}$ is defined in (5.26). $\boldsymbol{A}_m$ connects vectors $\mathbf{f}_m \sim \mathcal{N}(0, \boldsymbol{K}_{\mathbf{f}_m,\mathbf{f}_m})$ and $\boldsymbol{\epsilon}_m \sim \mathcal{N}(0, \sigma_m^2 \boldsymbol{I})$ of latent and noise process values to $\mathbf{v}$ as $\mathbf{f}_m + \boldsymbol{\epsilon}_m = \boldsymbol{A}_m \mathbf{v}$. Then the expected log likelihood of each factor may be expressed as

$$\begin{aligned} &\mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}}\left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}})\right] = \mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}}\left[p_{\boldsymbol{\theta}_m}(\mathbf{f}_m + \boldsymbol{\epsilon}_m)\Big|_{\mathbf{f}_m+\boldsymbol{\epsilon}_m=\boldsymbol{A}_m\mathbf{v}}\right] \\ &= -\frac{1}{2}\left((\boldsymbol{A}_m\boldsymbol{\mu}_{\mathbf{v}|\mathbf{y}})^T \hat{\boldsymbol{K}}^{-1}_{\mathbf{f}_m,\mathbf{f}_m} \boldsymbol{A}_m\boldsymbol{\mu}_{\mathbf{v}|\mathbf{y}} + \ \mathrm{tr}\left(\boldsymbol{A}_m\boldsymbol{\Sigma}_{\mathbf{v}|\mathbf{y}}\boldsymbol{A}_m^T\hat{\boldsymbol{K}}^{-1}_{\mathbf{f}_m,\mathbf{f}_m}\right) + \log\det \hat{\boldsymbol{K}}_{\mathbf{f}_m,\mathbf{f}_m}\right) \end{aligned} \tag{5.33}$$

where $\hat{\boldsymbol{K}}_{\mathbf{f}_m,\mathbf{f}_m} = \boldsymbol{K}_{\mathbf{f}_m,\mathbf{f}_m} + \sigma_m^2\boldsymbol{I}$. Within each loop of the Structural EM algorithm, kernel hyperparameters, noise variances σ_m^2, and parent weights $\boldsymbol{\Lambda}_{m,n}$ are selected to maximize (5.33). Optimization is performed through the L-BFGS algorithm [86].

5.8.6 Prediction in the Homogeneous AcyGP model

In this section, we show how to produce predictions of attributed at unobserved locations in the AcyGP model. Once an AcyGP is trained and $\boldsymbol{\theta}$ and $\mathcal{G}$ have been se-

lected, we will need to produce predictions of the environment for use in query-driven adaptive sampling. In addition, prediction is required to produce the distributions $p(\mathbf{v} \mid \mathbf{y}, \boldsymbol{\theta}^{(t,s)}, \mathcal{G}^{(t)})$ used on lines 3 and 4 of Algorithm 15.

Prediction of unobserved attributes $\mathbf{y}^*$ at inputs $\boldsymbol{X}^*$ is achieved using a conditional Gaussian distribution. Following (5.26), covariances between $\mathbf{y}$ and $\mathbf{y}^*$ may be constructed elementwise as

$$[\boldsymbol{K}_{\mathbf{y}_m^*,\mathbf{y}_n}]_{i,j} = \sum_{q=1}^{D_y} [\boldsymbol{B}^{-1}]_{m,q} \, [\boldsymbol{B}^{-1}]_{n,q} \, \left[k_q(\boldsymbol{x}_i^*, \boldsymbol{x}_j) + \sigma_m^2 \delta_{\boldsymbol{x}_i^*,\boldsymbol{x}_j}\right], \tag{5.34}$$

where $\delta_{\boldsymbol{x}_i^*,\boldsymbol{x}_j}$ is the Kronecker delta. Analogous expressions may be constructed for $\boldsymbol{K}_{\mathbf{y},\mathbf{y}}$ and $\boldsymbol{K}_{\mathbf{y}^*,\mathbf{y}^*}$. Then, the predictions satisfy $\mathbf{y}^* \,|\, \mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{y}^*|\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{y}^*|\mathbf{y}}\right)$ with

$$\boldsymbol{\mu}_{\mathbf{y}^*|\mathbf{y}} = \boldsymbol{K}_{\mathbf{y}^*,\mathbf{y}} \, \boldsymbol{K}_{\mathbf{y},\mathbf{y}}^{-1} \, \mathbf{y}, \quad \boldsymbol{\Sigma}_{\mathbf{y}^*|\mathbf{y}} = \boldsymbol{K}_{\mathbf{y}^*,\mathbf{y}^*} - \boldsymbol{K}_{\mathbf{y}^*,\mathbf{y}} \, \boldsymbol{K}_{\mathbf{y},\mathbf{y}}^{-1} \, \boldsymbol{K}_{\mathbf{y},\mathbf{y}^*}. \tag{5.35}$$

$\mathbf{v} \,|\, \mathbf{y}$ is computed in this manner using $\mathbf{y}^* = \mathbf{w}$, and adding $\mathbf{y}$ into the distribution with zero variance.

5.9 Qualitative Constraints in the AcyGP Model

We now describe more completely how qualitative constraints on structure and correlations are introduced in the homogeneous AcyGP model, and how they influence the training process within structural EM. Once we introduce the heterogeneous AcyGP model in the following section, we will generalize these ideas for heterogeneous attributes.

The AcyGP model makes dependencies between attributes explicit by including or excluding edges in a graphical model that encodes causal dependencies. When the model is constructed, the presence of an edge $\mathbf{y}_n \to \mathbf{y}_m$ states that $y_m(\boldsymbol{x})$ has a term $\boldsymbol{\Lambda}_{m,n} y_n(\boldsymbol{x})$ added to it. Whether or not direct dependency between attributes is introduced in the model is controlled by whether an edge exists, and the effect of an edge is controlled through the parameter $\boldsymbol{\Lambda}_{m,n}$. The qualitative statements that we

allow map to constraints on these two aspects of the model. We allow statements on the existence or non-existence of edges, which are enforced as constraints over whether an edge exists in the solved DAG model $\mathcal{G}$, and we allow statements of monotonicity and relative sizes of directed relationships, which are enforced as constraints over the parameter $\mathbf{\Lambda}_{m,n}$.

5.9.1 Edge Existence Constraints

An expert may have different ways of expressing qualitative knowledge of structure. In the simplest form, they may know that y_n is a parent of y_m, like how temperature was a parent of density in our motivating example. They may also may not know whether y_m and y_n are causally related, but that if they are, then y_n is definitely the parent. We express these kinds of constraints through the previously mentioned sets of allowable parent sets, $\{\Xi_m\}$. A specification of Ξ_m is provided as input to the training process, and any parent set for y_m that does not exist in Ξ_m is not allowed in the DAG $\mathcal{G}$ that is solved in structural EM. For example, if y_n is definitely a parent of y_m, then all parent sets $\Pi_m^{\mathcal{G}}$ in Ξ_m must contain the index n as an element. If y_n is never a parent of y_m, then no $\Pi_m^{\mathcal{G}}$ in Ξ_m contains the index n. If y_n is only ever a parent of y_m when y_l is also a parent, then all $\Pi_m^{\mathcal{G}}$ in Ξ_m that contains l also contains n.

When a user constructs an AcyGP and begins training, they may specify the sets $\{\Xi_m\}$ manually if they wish, or they may construct a function $allow(\Pi_m^{\mathcal{G}})$ that returns True when $\Pi_m^{\mathcal{G}} \in \Xi_m$, which allows greatest flexibility. However, this is often tedious, and more general than is usually needed. A user can also specify a set of known edges that must exist in the model $\mathcal{E}_{exist}$ and a set of edges that must not exist in the model $\mathcal{E}_{not-exist}$. Then $allow(\Pi_m^{\mathcal{G}})$ can be computed as

$$allow(\Pi_m^{\mathcal{G}}) = \begin{cases} \text{True}, & n \in \Pi_m^{\mathcal{G}} \ \forall n \text{ s.t. } (\mathbf{y}_n \to \mathbf{y}_m) \in \mathcal{E}_{exist} \text{ and } l \notin \\ & \Pi_m^{\mathcal{G}} \ \forall l \text{ s.t. } (\mathbf{y}_l \to \mathbf{y}_m) \in \mathcal{E}_{not-exist} \\ \text{False}, & \text{otherwise.} \end{cases} \tag{5.36}$$

Users can also specify that $\mathcal{E}_{exist}$ is simply the complement set of $\mathcal{E}_{not-exist}$ or vice versa, if they wish to merely exclude certain edges.

Within structural EM, the sets Ξ_m are used when generating candidate DAG structures. Structures are selected on line 4 of Algorithm 15 that maximize

$$\mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}}\left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}})\right] - \frac{|\Pi_m^{\mathcal{G}}|}{2}\log N.$$

In a naive implementation, for each m, the sets $\Pi_m^{\mathcal{G}}$ in Ξ_m are enumerated. Every possible parent set is generated, and if $allow(\Pi_m^{\mathcal{G}})$ is True, the optimal parameters for that parent set are found. L-BFGS is used to perform gradient ascent to find the solution to

$$\max_{\boldsymbol{\theta}_m} \mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}}\left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}})\right] - \frac{|\Pi_m^{\mathcal{G}}|}{2}\log N$$

for every possible $\Pi_m^{\mathcal{G}}$ in Ξ_m. Determining $\mathcal{G}^{(t+1)}$ is then achieved by selecting a single parent set for each attribute, out of those that were evaluated, that result in an acyclic structure and the largest sum of the expressions above at the optimized values of $\boldsymbol{\theta}_m$. There exist many structure learning algorithms that take rewards for edges and solve for the maximum reward DAG, and we use an A* based approach [156].

5.9.2 Edge Strength Constraints

The expert may also specify constraints on monotonicity and relative sizes of specific edges, when they exist. These translate to constraints on the parameters $\boldsymbol{\Lambda}_{m,n}$.

Specifically, we allow an expert to specify that when y_n is a parent of y_m, that the expectation of y_m either increases or decreases with the value of y_n. As an example, we saw that density is expected to decrease when temperature increases, so this is a monotonically decreasing relationship. Since the homogeneous AcyGP model is a linear model, the parameter $\boldsymbol{\Lambda}_{m,n}$ is interpreted as the strength of the effect of $y_n(\boldsymbol{x})$ on $y_m(\boldsymbol{x})$. To assert that y_m monotonically increases in expectation with increasing y_n, we require that $\boldsymbol{\Lambda}_{m,n} > 0$, and to specify that $y_m(\boldsymbol{x})$ monotonically decreases in expectation with increasing y_n, we require that $\boldsymbol{\Lambda}_{m,n} < 0$.

We can also specify that the effect of y_n on y_m, if it exists, is small. This can be achieved by specifying that the parameter $\mathbf{\Lambda}_{m,n}$ is small, so that $-\epsilon < \mathbf{\Lambda}_{m,n} < \epsilon$ for a small value ϵ.

The expert specifies these constraints as a list of statements that y_m monotonically increases/decreases with y_n, and that the effect of y_n on y_m is small, that are given when the AcyGP is constructed. These statements are then converted into constraints to be used when the AcyGP is trained. They are used whenever the parameters of the AcyGP are solved, which occurs in lines 3 and 4 of structural EM. These constraints can be passed directly into gradient optimization algorithm, in this case, L-BFGS. L-BFGS is used to find a value of $\boldsymbol{\theta}$ that satisfies these constraints, so that the qualitative statements given by the user necessarily hold.

5.10 The Heterogeneous AcyGP Model

In this section we generalized the AcyGP model beyond continuous unbounded attributes, by using likelihood approximations developed for heterogeneous Gaussian processes. The homogeneous AcyGP model is powerful, and permits exact expressions for likelihood and posterior predictive distributions. However, in many adaptive sampling domains, we wish to use the AcyGP model to consider non-Gaussian attributes, such as categorical variables or continuous valued variables with known minima or maxima. For example, the presence or absence of a seep at location $\boldsymbol{x}$ may be described by the function $y(\boldsymbol{x})$, but the output of $y(\boldsymbol{x})$ is Boolean variable. We still wish to capture spatio-temporal correlations and correlations between observables through an interpretable DAG model, while being able to handle different types of attributes.

We model these environments through the *heterogeneous AcyGP model*. The model is heterogeneous because it allows each attribute to follow a different type of distribution. Our approach combines ideas from heterogeneous Gaussian process regression presented by Moreno-Muñoz et al. [96] with the homogeneous AcyGP model. The primary difference from the homogeneous AcyGP model is more complex depen-

dencies between attributes in order to model non-Gaussian attributes. Likelihood computation and prediction cannot be performed in closed form, and we rely instead on a variational approximation to inference. Compared to [96], we introduce connections between attributes that depend directly on non-Gaussian attributes, and show how approximate inference may be performed with more complex quadrature rules.

5.10.1 Heterogeneous AcyGP Construction

In this section we describe how a heterogeneous AcyGP is constructed. As in the homogeneous model, we enforce a DAG structure $\mathcal{G}$ between D_y attributes in the multi-output Gaussian process. In the heterogeneous model, each attribute is assumed to be drawn from a specified type of distribution, such as a Gaussian distribution or a categorical distribution. These distributions are parameterized, and the heterogeneous AcyGP model uses latent Gaussian processes to model a subset of the parameters as a function of input variables. For example, a Gaussian attribute may be parameterized by a input-dependent mean and a constant variance, and the mean of the distribution is drawn from a latent Gaussian process. In this way, the attribute distributions vary with the inputs through the latent processes.

Each attribute y_m is modeled using an *attribute model* that describes how a distribution for $y_m(\boldsymbol{x})$ is constructed from input-dependent parameters. Formally, an attribute model for $y_m(\boldsymbol{x})$ in the heterogeneous AcyGP model is defined to be a tuple $(J_m, p_{\boldsymbol{\beta}_m}, t_{\boldsymbol{\lambda}_{l,m,j}})$, where $p_{\boldsymbol{\beta}_m}$ is a probability distribution used to predict $y_m(\boldsymbol{x})$) from input dependent parameters, J_m is an integer representing how many latent processes are needed to model $p_{\boldsymbol{\beta}_m}$, and $t_{\boldsymbol{\lambda}_{l,m,j}}$ is a function that describes how y_m should be modeled to affect child attributes. Formally, the attribute model represents that $y_m(\boldsymbol{x})$ is distributed as

$$y_m(\boldsymbol{x}) \sim p_{\boldsymbol{\beta}_m}(y_m(\boldsymbol{x}) \mid \boldsymbol{\alpha}_m(\boldsymbol{x})). \tag{5.37}$$

The distribution depends on a vector $\boldsymbol{\alpha}_m(\boldsymbol{x}) = [\alpha_{m,1}(\boldsymbol{x}), \ldots, \alpha_{m,J_m}(\boldsymbol{x})]^T$ of J_m real-valued parameters that vary with input $\boldsymbol{x}$, and a vector $\boldsymbol{\beta}_m$ of constant parameters.

For example, if $y_m(\boldsymbol{x})$ is Gaussian distributed with input-dependent mean and constant variance, then $\alpha_1(\boldsymbol{x})$ would be the input-dependent mean, $\beta_{m,1}$ would be the constant variance, $p_{\boldsymbol{\beta}_m}(y_m(\boldsymbol{x}) \mid \boldsymbol{\alpha}_m(\boldsymbol{x})) = \mathcal{N}(\alpha_{m,1}(\boldsymbol{x}), \beta_{m,1})$, and since only one input parameter is required, $J_m = 1$.

The input-dependent parameter $\alpha_{m,j}(\boldsymbol{x})$ is constructed as the sum of a latent GP $f_{m,j}(\boldsymbol{x}) \sim \mathcal{GP}(0, k_{m,j})$ and the functions $t_{\boldsymbol{\lambda}_{m,n,j}}$ as

$$\alpha_{m,j}(\boldsymbol{x}) = f_{m,j}(\boldsymbol{x}) + \sum_{n \in \Pi_m^{\mathcal{G}}} t_{\boldsymbol{\lambda}_{m,n,j}}(y_n(\boldsymbol{x})). \tag{5.38}$$

In the example of a Gaussian y_m, we model that parameters of child attributes $\boldsymbol{\alpha}_l$ scale linearly with y_m, so $t_{\boldsymbol{\lambda}_{l,m,j}}(y_m(\boldsymbol{x})) = \lambda_{l,m,j,1}\ y_m(\boldsymbol{x})$. The number of latent processes J_m and the number of parameters in $\boldsymbol{\beta}_m$ is specified by the type of distribution of y_m.

The function $t_{\boldsymbol{\lambda}_{l,m,j}}$ describes how the value of y_m influences the parameters $\boldsymbol{\alpha}_l$ of a child attribute y_l. For example, to encode a positive correlation between y_l and y_m, $t_{\boldsymbol{\lambda}_{l,m,j}}(y_m(\boldsymbol{x}))$ should return a larger value for larger values of $y_m(\boldsymbol{x})$, and the function $p_{\boldsymbol{\beta}}(y_l(\boldsymbol{x}) \mid \boldsymbol{\alpha}_l(\boldsymbol{x}))$ should make larger values of $y_l(\boldsymbol{x})$ more likely with larger $\boldsymbol{\alpha}_l(\boldsymbol{x})$. The form of the function $t_{\boldsymbol{\lambda}_{l,m,j}}(y_,(\boldsymbol{x}))$ and the number of parameters $\boldsymbol{\lambda}_{l,m,j}$ is specified by the type of distribution of the *parent attribute* $y_m(\boldsymbol{x})$. For example, if y_m is modeled as a categorical distribution with Gaussian parent y_n, the effect of $y_n(\boldsymbol{x})$ on $\alpha_{m,j}(\boldsymbol{x})$ is the previously mentioned linear scaling rule $\lambda_{m,n,j,1}\ y_n(\boldsymbol{x})$ for Gaussian attribute models.

When an AcyGP is trained, structure and parameters will be optimized like in the homogeneous AcyGP model. In addition to kernel parameters, training a heterogeneous AcyGP will optimize the parameters $\boldsymbol{\beta}_m$ and $\boldsymbol{\lambda}_{l,m,j}$.

To assist in solving for AcyGP structure, we wish to ensure that an AcyGP with structure $\mathcal{G}$ models a broader set of distributions than an AcyGP with structure $\mathcal{G}'$ with a subset of edges from $\mathcal{G}$. To enforce this requirement, we assert that $t_{\boldsymbol{\lambda}_{l,m,j}}$ must be formulated so that there exists values of $\boldsymbol{\lambda}_{l,m,j}$ for which $t_{\boldsymbol{\lambda}_{l,m,j}}(y_m(\boldsymbol{x})) = 0$ for all $y_m(\boldsymbol{x})$. In this way, the effect of y_m can always be removed for its children with a choice of the parameters $\boldsymbol{\lambda}_{l,m,j}$, so that an AcyGP with structure $\mathcal{G}$ will model

Table 5.1: Common attribute models for use in the heterogeneous AcyGP model.

Data Domain	Distribution	J	$p_{\boldsymbol{\beta}_m}$	$t_{\boldsymbol{\lambda}_{l,m,j}}$
$\mathbb{R}$	Normal	1	$\mathcal{N}(\alpha_{m,1}, \beta_{m,1})$	$\lambda_{l,m,j,1}\, y_m$
$\mathbb{R}_{\geq 0}$	Gamma	1	$G(e^{\alpha_{m,1}}, \beta_{m,1})$	$\lambda_{l,m,j,1}\, y_m$
$[0,1]$	Beta	1	$B(\beta_{m,1}\, \sigma(\alpha_{m,1})_1, \beta_{m,1}\, \sigma(\alpha_{m,1})_2)$	$\lambda_{l,m,j,1}\, y_m$
$\{1, \ldots, C\}$	Categorical	$C-1$	$\sigma(\boldsymbol{\alpha}_m)_{y_m}$	λ_{l,m,j,y_m}

a strictly larger set of distributions than an AcyGP with structure $\mathcal{G}'$. For example, if y_m follows a Gaussian attribute model, then $t_{\boldsymbol{\lambda}_{l,m,j}}(y_m(\boldsymbol{x})) = 0$ when $\lambda_{l,m,j,1} = 0$. In this way, an AcyGP with edge $\mathbf{y}_m \rightarrow \mathbf{y}_l$ can capture any distribution that can be represented by an AcyGP without this edge by selecting $\lambda_{l,m,j,1} = 0$.

Table 5.1 shows common attribute models used in the heterogeneous AcyGP model. The choice of attribute model is based on the domain of the attribute y_m, such as the set of positive numbers or the set $[0,1]$, so that the choice of distribution produces predictions within that domain. Domains that are scaled or offset from those listed in Table 5.1 can be modeled by applying the reverse transformation to the data prior to modeling. $\mathcal{N}$, G, and B are used to represent the probability density functions for normal, gamma, and beta distributions,

$$\mathcal{N}(a,b) = \frac{1}{\sqrt{2\pi b}} \exp\left(-\frac{(y_m - a)^2}{2b}\right) \tag{5.39}$$

$$G(a,b) = \frac{1}{\Gamma(a)\, b^a}\, y_m^{a-1}\, e^{-\frac{y_m}{b}} \tag{5.40}$$

$$B(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\,\Gamma(b)}\, y_m^{a-1}\, (1-y_m)^{b-1}\,, \tag{5.41}$$

and $\sigma(\boldsymbol{a})_b$ is the softmax function,

$$\sigma(\boldsymbol{a})_b = \begin{cases} \frac{\exp(a_i)}{1+\sum_{i=1}^{|\boldsymbol{a}|}\exp(a_i)}, & b \in \{1, \ldots, |\boldsymbol{a}|\} \\ \frac{1}{1+\sum_{i=1}^{|\boldsymbol{a}|}\exp(a_i)}, & b = |\boldsymbol{a}| + 1 \end{cases}. \tag{5.42}$$

The following example shows how attribute models are combined for a given DAG structure.

Example 2. *Consider an AcyGP with 3 attributes, y_1, y_2, and y_3. $y_1(\boldsymbol{x})$ is a Gaussian random variable, $y_2(\boldsymbol{x})$ is a categorical random variable with domain $\{1, 2\}$, and $y_3(\boldsymbol{x})$ is a categorical random variable with domain $\{1, 2, 3\}$. The DAG structure has two edges, $y_1 \to y_3$ and $y_2 \to y_3$.*

The mean of $y_1(\boldsymbol{x})$ is modeled as $\alpha_{1,1}(\boldsymbol{x})$, and its variance is specified by a static parameter $\beta_{1,1}$. Since y_1 has no parents, $\alpha_{1,1}(\boldsymbol{x}) = f_{1,1}(\boldsymbol{x})$ for the latent GP $f_{1,1}$.

Similarly, $y_2(\boldsymbol{x})$ depends on a single input-dependent parameter $\alpha_{2,1}(\boldsymbol{x})$ and no static parameters. y_2 has no parents, so $\alpha_{2,1}(\boldsymbol{x}) = f_{2,1}(\boldsymbol{x})$ for GP $f_{2,1}$.

Finally, $y_3(\boldsymbol{x})$ depends on two input-dependent parameters, $\alpha_{3,1}(\boldsymbol{x})$ and $\alpha_{3,2}(\boldsymbol{x})$ and no static parameters. y_3 has y_1 and y_2 as parents, so

$$\alpha_{3,1}(\boldsymbol{x}) = f_{3,1}(\boldsymbol{x}) + \lambda_{3,1,1,1}\, y_1(\boldsymbol{x}) + \lambda_{3,2,1,1}\, \delta_{y_2(\boldsymbol{x}),1} + \lambda_{3,2,1,2}\, \delta_{y_2(\boldsymbol{x}),2}$$
$$\alpha_{3,2}(\boldsymbol{x}) = f_{3,2}(\boldsymbol{x}) + \lambda_{3,1,2,1}\, y_1(\boldsymbol{x}) + \lambda_{3,2,2,1}\, \delta_{y_2(\boldsymbol{x}),1} + \lambda_{3,2,2,2}\, \delta_{y_2(\boldsymbol{x}),2}$$

for GPs $f_{3,1}$ and $f_{3,2}$. In analogy to the homogeneous AcyGP model, the parameters depend linearly on the value of $y_1(\boldsymbol{x})$, with $|\boldsymbol{\lambda}_{3,1,1}| = 1$. The dependence on $y_2(\boldsymbol{x})$ is more complex, and uses multiple parameters $|\boldsymbol{\lambda}_{3,2,1}| = 2$ to encode the distinct influences of $y_2(\boldsymbol{x}) = 1$ and $y_2(\boldsymbol{x}) = 2$.

Through this definition, probability distributions for $y_m(\boldsymbol{X})$ depend directly on the parents $y_{\Pi_m^{\mathcal{G}}}(\boldsymbol{X})$. This introduces a structural equation model connecting the attributes, so that the data factors according to the DAG $\mathcal{G}$. Spatio-temporal correlations are captured by the Gaussian processes $f_{m,j}$, which correlates spatially close observations by making similar values of latent functions more likely.

In Moreno-Muñoz et al. [96], the parameters $\{\boldsymbol{\alpha}_m\}$ depend only on mixtures of latent processes $\{f_{m,j}\}$, rather than the true attributes $\{y_m\}$. This means that for $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ such that $\boldsymbol{y}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{x}_i) = \boldsymbol{y}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{x}_j)$, the effect on prediction of $y_m(\boldsymbol{x}_i)$ and $y_m(\boldsymbol{x}_j)$ may be different, because it is not guaranteed that $\boldsymbol{f}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{x}_i) = \boldsymbol{f}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{x}_j)$. In contrast, our approach treats the attributes as a more fundamental quantity, and depends on $\boldsymbol{y}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{x}_i)$ directly, eliminating this problem.

5.10.2 Modeling Qualitative Constraints

Expert knowledge of interactions in the environment helps to constrain model parameters, leading to a more accurate model and faster selection of optimized parameters. We have already discussed that experts may possess incomplete knowledge of the interactions between variables in the environment, but they also may possess partial knowledge of the nature of interactions between specific variables, if they exist. This incomplete knowledge could take many forms, but we specifically consider here qualitative relationships between variables that specify whether correlations are positive or negative.

Formulation of a structural equation model for AcyGPs makes it simple to express a few key qualitative relationships between variables as restrictions on $\{\boldsymbol{\lambda}_{m,n,j}\}$. For an inter-variable relationship expressed as an edge $y_n \to y_m$, we may express that continuous y_m increases in expectation with increased continuous y_n or when categorical $y_n = i$ as

$$\begin{aligned} &\lambda_{m,n,1,1} > 0, \quad \text{continuous } y_m, \text{ continuous } y_n \\ &\lambda_{m,n,1,i} > 0, \quad \text{continuous } y_m, \text{ categorical } y_n = i. \end{aligned} \tag{5.43}$$

We may similarly express that the probability of categorical $y_n = j$ increases with increased continuous y_n or when categorical $y_n = i$ as

$$\begin{aligned} &\lambda_{m,n,j,1} > \lambda_{m,n,j',1} \quad \forall\ j' \neq j, \quad \text{categorical } y_m = j, j < C, \text{ continuous } y_n \\ &\lambda_{m,n,j',1} < 0 \qquad\quad \forall\ j', \qquad\quad \text{categorical } y_m = C, \text{ continuous } y_n \\ &\lambda_{m,n,j,i} > \lambda_{m,n,j',i} \quad \forall\ j' \neq j, \quad \text{categorical } y_m = j, j < C, \text{ categorical } y_n = i \\ &\lambda_{m,n,j',i} < 0 \qquad\quad \forall\ j', \qquad\quad \text{categorical } y_m = C, \text{ categorical } y_n = i. \end{aligned} \tag{5.44}$$

Opposite statements, that continuous y_m decreases in expectation or the probability of categorical $y_n = j$ decreases when continuous y_n increases or when categorical $y_n = i$, can be achieved by reversing the signs of all inequalities.

5.10.3 Selection of Parameters and DAG Structure in the Heterogeneous AcyGP Model

In this section, we describe how parameters and DAG structure are selected in the heterogeneous AcyGP model. Since the model has been generalized and includes additional parameters compared to the homogeneous AcyGP model, the form of the objective must also be modified.

Selection of parameters for a given structure and choice of structure is still performed through optimization of BIC. Like in the homogeneous AcyGP model, missing observations of an attribute at an input where another attribute has been observed means optimization must be done through Structural EM, with vectors of observed and unobserved values $\mathbf{v}_m$.

For fixed kernels and attribute models, the number of kernel parameters and parameters in $\boldsymbol{\beta}_m$ does not change with structure. However, the number of parent dependence parameters $\boldsymbol{\lambda}_{m,n,j}$ now depends on the choices of parents. For a Gaussian parent y_n, only a single parameter is needed in $\boldsymbol{\lambda}_{m,n,j}$, but for a categorical parent with domain size C, $C-1$ parameters are needed in $\boldsymbol{\lambda}_{m,n,j}$. In addition, these edge parameters exist for each of the J_m latent processes that are used in modeling y_m. As a result, the number of parameters in a heterogeneous AcyGP model associated with attribute m is $J_m \left(\sum_{n \in \Pi_m^{\mathcal{G}}} |\boldsymbol{\lambda}_{m,n,j}|\right)$, and the objective function that is optimized in structural EM becomes

$$\sum_{m=1}^{D_y} \mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}} \left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \mid \mathbf{v}_{\Pi_m^{\mathcal{G}}})\right] - \frac{J_m}{2} \left(\sum_{n \in \Pi_m^{\mathcal{G}}} |\boldsymbol{\lambda}_{m,n,j}|\right) \log N. \tag{5.45}$$

Here, the vector $\boldsymbol{\theta}_m$ includes kernel hyperparameters and all parameters $\boldsymbol{\beta}_m$ and $\boldsymbol{\lambda}_{m,n,j}$.

Changing the parameter count does not change the way that structural EM operates in any meaningful way. However, the expectation $\mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}} \left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \mid \mathbf{v}_{\Pi_m^{\mathcal{G}}})\right]$ no longer has a closed form solution, and we must resort to approximations that we describe in the following sections.

5.10.4 Optimization Procedure for a Candidate Structure

Combining multiple non-Gaussian distributions in the heterogeneous AcyGP model means that posterior distributions conditioned on observations can no longer be written in closed form, and it is difficult to even describe the posterior distributions exactly. This makes it difficult to compute the likelihood of observations. In this section, we describe how to approximate likelihood using a variational technique that was developed by Moreno-Muñoz et al. [96], which we adapt to the heterogeneous AcyGP model.

To see why inference and likelihood computation is difficult, consider an AcyGP with a single categorical attribute with domain $\{1, 2\}$ and a single observation $y_1(\boldsymbol{x}_i)$. The posterior distribution of $f_{1,1}(\boldsymbol{x}_i)$ is given as

$$\begin{aligned} p(f_{1,1}(\boldsymbol{x}_i) \mid y_1(\boldsymbol{x}_i)) &\propto p(y_1(\boldsymbol{x}_i) \mid f_{1,1}(\boldsymbol{x}_i))\ p(f_{1,1}(\boldsymbol{x}_i)) \\ &= \sigma(f_{1,1}(\boldsymbol{x}_i))_{y_m(\boldsymbol{x}_i)}\ p(f_{1,1}(\boldsymbol{x}_i)). \end{aligned}$$

This posterior is difficult to work with because of the sigmoid term. The distribution is non-Gaussian, and a normalization constant cannot be computed in closed form. Prediction of any $y_1(\boldsymbol{x}_j) \mid y_1(\boldsymbol{x}_i)$ must be computed through integration over $f_{1,1}(\boldsymbol{x}_j) \mid y_1(\boldsymbol{x}_i)$, which is similarly non-Gaussian, and it is not clear how to efficiently sample from or perform quadrature over this distribution. The difficulty is only exacerbated in the case of additional observations.

Instead, we follow the approach of Moreno-Muñoz et al. [96] and perform inference and training through the use of a variational approximation. Moreno-Muñoz et al. developed this method to model heterogeneous Gaussian processes, in which all parameters are modeled using a fully correlated linear MOGP. The variational approach optimizes likelihood using an approximation to the posterior distribution of the latent Gaussian processes. The posterior distribution of the GPs are modeled as Gaussian distributed, and a lower bound on likelihood is developed for any Gaussian posterior. This lower bound can be approximated using Gauss-Hermite quadrature, and the Gaussian posterior that results in the largest lower bound on likelihood is

solved. The optimized value of the bound is used in structural EM for training, and this variational posterior is used for inference and prediction. The variational approximation can be more readily computed than the exact posterior, is faster for large data sets, and can handle cases for which no analytic expression for the posterior distribution exists.

In the variational approach, the processes $f_{m,j}$ are modeled at a vector of inducing locations $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{N_z}]$. In general, fewer inducing locations will be chosen than training inputs, so that training will be accelerated. We denote the values of the processes at the inducing locations as $\mathbf{u}_{m,j} = [f_{m,j}(\boldsymbol{z}_1), \ldots, f_{m,j}(\boldsymbol{z}_{N_z})]^T$.

We then make the variational choice to model the posterior distributions on the latent processes values as the closest Gaussian distributions to the true posteriors. We still perform structural EM, so we compute posteriors with respect to the vector of all parent observables $\mathbf{v}_{\Pi_m^{\mathcal{G}}}$. The approximate posterior distribution is denoted $q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j}) = \mathcal{N}(\boldsymbol{\mu}_{m,j}, \boldsymbol{S}_{m,j})$. $q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j})$ is an approximation to the true (non-Gaussian) posterior $p_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j} \mid \mathbf{v}_{\Pi_m^{\mathcal{G}}})$, but we will solve for $\boldsymbol{\mu}_{m,j}$ and $\boldsymbol{S}_{m,j}$ so that $q_{\boldsymbol{\theta}_{m,j}}$ is close to $p_{\boldsymbol{\theta}_{m,j}}$. The combined posterior of $\mathbf{f}_{m,j}, \mathbf{u}_{m,j}$ is approximated using the exact conditional distribution of $\mathbf{f}_{m,j} \mid \mathbf{u}_{m,j}$ and the variational distribution of $\mathbf{u}_{m,j}$ as

$$\begin{aligned} p_{\boldsymbol{\theta}_{m,j}}(\mathbf{f}_{m,j}, \mathbf{u}_{m,j} \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) &\approx q_{\boldsymbol{\theta}_{m,j}}(\mathbf{f}_{m,j}, \mathbf{u}_{m,j}) \\ &= p_{\boldsymbol{\theta}_{m,j}}(\mathbf{f}_{m,j} \,|\, \mathbf{u}_{m,j})\; q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j}). \end{aligned} \tag{5.46}$$

Here, $p_{\boldsymbol{\theta}_{m,j}}(\mathbf{f}_{m,j} \,|\, \mathbf{u}_{m,j})$ is computed as the predictive distribution of a Gaussian process at inputs $\boldsymbol{X}$ conditioned on observations of $\mathbf{u}_{m,j}$ at inputs $\boldsymbol{Z}$.

We denote the complete vectors of all GP values for attribute index m as $\mathbf{f}_m^T = [\mathbf{f}_{m,1}^T, \ldots, \mathbf{f}_{m,J_m}^T]$ and $\mathbf{u}_m^T = [\mathbf{u}_{m,1}^T, \ldots, \mathbf{u}_{m,J_m}^T]$. Since the processes $f_{m,1}, \ldots, f_{m,j}$ are

mutually independent, we may compute joint distributions as

$$q_{\boldsymbol{\theta}_m}(\mathbf{u}_m) = \prod_{j=1}^{J_m} q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j}) \tag{5.47}$$

$$p_{\boldsymbol{\theta}_m}(\mathbf{f}_m \,|\, \mathbf{u}_m) = \prod_{j=1}^{J_m} p_{\boldsymbol{\theta}_{m,j}}(\mathbf{f}_{m,j} \,|\, \mathbf{u}_{m,j}), \tag{5.48}$$

which jointly imply that

$$q_{\boldsymbol{\theta}_m}(\mathbf{f}_m, \mathbf{u}_m) = p_{\boldsymbol{\theta}_m}(\mathbf{f}_m \,|\, \mathbf{u}_m)\; q_{\boldsymbol{\theta}_m}(\mathbf{u}_m). \tag{5.49}$$

Using the variational distribution, a lower bound for the log likelihood is derived using Jensen's inequality, which implies that for random variable $\mathbf{r}$, $\log \mathbb{E}\left[g(\mathbf{r})\right] \geq \mathbb{E}\left[\log g(\mathbf{r})\right]$. Algebraic manipulation of the log likelihood and application of Jensen's inequality leads to

$$\begin{aligned}
&\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) \\
&\quad = \log \int p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathbf{f}_m)\; p_{\boldsymbol{\theta}_m}(\mathbf{f}_m \,|\, \mathbf{u}_m)\; p_{\boldsymbol{\theta}_m}(\mathbf{u}_m)\, d\mathbf{f}_m\; d\mathbf{u}_m \\
&\quad = \log \int q_{\boldsymbol{\theta}_m}(\mathbf{f}_m, \mathbf{u}_m) \frac{p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathbf{f}_m)\; p_{\boldsymbol{\theta}_m}(\mathbf{f}_m \,|\, \mathbf{u}_m)\; p_{\boldsymbol{\theta}_m}(\mathbf{u}_m)}{q_{\boldsymbol{\theta}_m}(\mathbf{f}_m, \mathbf{u}_m)}\; d\mathbf{f}_m\; d\mathbf{u}_m \\
&\quad = \log \mathbb{E}_{q_{\boldsymbol{\theta}_m}(\mathbf{f}_m, \mathbf{u}_m)} \left[\frac{p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathbf{f}_m)\; p_{\boldsymbol{\theta}_m}(\mathbf{f}_m \,|\, \mathbf{u}_m)\; p_{\boldsymbol{\theta}_m}(\mathbf{u}_m)}{q_{\boldsymbol{\theta}_m}(\mathbf{f}_m, \mathbf{u}_m)} \right] \\
&\quad \geq \mathbb{E}_{q_{\boldsymbol{\theta}_m}(\mathbf{f}_m, \mathbf{u}_m)} \left[\log \frac{p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathbf{f}_m)\; p_{\boldsymbol{\theta}_m}(\mathbf{u}_m)}{q_{\boldsymbol{\theta}_m}(\mathbf{u}_m)} \right] \\
&\quad = \mathbb{E}_{q_{\boldsymbol{\theta}_m}(\mathbf{f}_m)} \left[\log p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathbf{f}_m) \right] - \mathrm{KL}(q_{\boldsymbol{\theta}_m}(\mathbf{u}_m) \;||\; p_{\boldsymbol{\theta}_m}(\mathbf{u}_m)) \\
&\quad = \sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\theta}_m}(\boldsymbol{f}_m(\boldsymbol{x}_i))} \left[\log p_{\boldsymbol{\theta}_m}(y_m(\boldsymbol{x}_i) \,|\, \boldsymbol{y}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{x}_i), \boldsymbol{f}_m(\boldsymbol{x}_i)) \right] \\
&\qquad - \sum_{j=1}^{J_m} \mathrm{KL}(q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j}) \;||\; p_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j})),
\end{aligned} \tag{5.50}$$

where $\mathrm{KL}(\cdot \;||\; \cdot)$ denotes the Kullback-Leibler divergence between two distributions.

The distribution $q_{\boldsymbol{\theta}_m}(\mathbf{f}_m)$ is computed as

$$\begin{aligned} q_{\boldsymbol{\theta}_m}(\mathbf{f}_m) &\triangleq \int p_{\boldsymbol{\theta}_m}(\mathbf{f}_m \,|\, \mathbf{u}_m)\; q_{\boldsymbol{\theta}_m}(\mathbf{u}_m)\; d\mathbf{u}_m \\ &= \prod_{j=1}^{J_m} \int p_{\boldsymbol{\theta}_{m,j}}(\mathbf{f}_{m,j} \,|\, \mathbf{u}_{m,j})\; q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j})\; d\mathbf{u}_{m,j} \\ &= \prod_{j=1}^{J_m} \mathcal{N}\Big(\boldsymbol{K}_{\mathbf{f}_{m,j},\mathbf{u}_{m,j}} \boldsymbol{K}^{-1}_{\mathbf{u}_{m,j},\mathbf{u}_{m,j}} \boldsymbol{\mu}_{m,j}, \\ &\qquad\qquad \boldsymbol{K}_{\mathbf{f}_{m,j},\mathbf{u}_{m,j}} \boldsymbol{K}^{-1}_{\mathbf{u}_{m,j},\mathbf{u}_{m,j}} \left(\boldsymbol{S}_{m,j} - \mathbf{K}_{\mathbf{u}_{m,j},\mathbf{u}_{m,j}}\right) \boldsymbol{K}^{-1}_{\mathbf{u}_{m,j},\mathbf{u}_{m,j}} \boldsymbol{K}_{\mathbf{u}_{m,j},\mathbf{f}_{m,j}}\Big). \end{aligned} \tag{5.51}$$

The variational strategy is to maximize the lower bound given in (5.50) through selection of the AcyGP parameters $\boldsymbol{\theta}$ and $\mathcal{G}$ in addition to the parameterization of $q_{\boldsymbol{\theta}_m}(\mathbf{u}_m)$ (through the values of $\boldsymbol{\mu}_m$ and $\boldsymbol{S}_m$). It may be verified by substitution that for $q_{\boldsymbol{\theta}_m}(\mathbf{f}_m, \mathbf{u}_m) = p_{\boldsymbol{\theta}_m}(\mathbf{f}_m, \mathbf{u}_m \,|\, \mathbf{v}_{\Pi^{\mathcal{G}}_m})$ the bound is tight. This means the optimization is exact for AcyGPs with Gaussian attributes with $\boldsymbol{Z} = \boldsymbol{X}$, because the true posterior distribution is Gaussian, matching the variational assumption. When the true posterior distribution $p_{\boldsymbol{\theta}_m}(\mathbf{f}_m, \mathbf{u}_m \,|\, \mathbf{v}_{\Pi^{\mathcal{G}}_m})$ is not Gaussian distributed, variational optimization solves for the Gaussian distribution with the closest likelihood to the true posterior.

Applying this form to the Structural EM objective given in (5.45), we reach a final objective to be optimized of

$$\begin{aligned} \sum_{m=1}^{D_y} \Bigg\{ &\mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}} \left[\sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\theta}_m}(\boldsymbol{f}_m(\boldsymbol{x}_i))} \left[\log p_{\boldsymbol{\theta}_m}(y_m(\boldsymbol{x}_i) \,|\, \boldsymbol{y}_{\Pi^{\mathcal{G}}_m}(\boldsymbol{x}_i), \boldsymbol{f}_m(\boldsymbol{x}_i)) \right] \right] \\ &- \sum_{j=1}^{J_m} \mathrm{KL}(q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j}) \,||\, p_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j})) - \frac{J_m}{2} \left(\sum_{n \in \Pi^{\mathcal{G}}_m} |\boldsymbol{\lambda}_{m,n,j}| \right) \log N \Bigg\}. \end{aligned} \tag{5.52}$$

5.10.5 Approximate Computation Through Numerical Quadrature

For heterogeneous attributes, evaluation of the expectations in (5.52) over the distributions of $\mathbf{v} \,|\, \mathbf{y}, \boldsymbol{\theta}^{(t,s)}, \mathcal{G}^{(t)}$ and $q_{\boldsymbol{\theta}_m}(\boldsymbol{f}_m(\boldsymbol{x}_i))$ typically lack closed form solutions. Precise evaluation can be achieved through Monte Carlo techniques, but Monte Carlo estimation is time consuming. Evaluation of the expectations is therefore evaluated through the use of quadrature rules that we describe in this section. The basics of numerical quadrature were covered in Section 5.4.

Gaussian expectations are maximized through Gauss-Hermite quadrature, so we use this method for approximation expectations of $q_{\boldsymbol{\theta}_m}(\boldsymbol{f}_m(\boldsymbol{x}_i))$. To model $\mathbf{v} \,|\, \mathbf{y}, \boldsymbol{\theta}^{(t,s)}, \mathcal{G}^{(t)}$, we compute a quadrature rule $\mathcal{Q}[p_{\boldsymbol{\theta}_m^{(t,s)}}(\boldsymbol{y}(\boldsymbol{x}_i) \,|\, \mathbf{y}, \mathcal{G}^{(t)})]$ for each $\boldsymbol{x}_i$. The method to construct this quadrature rule is given in the next section.

(5.52) is computed using quadrature rules $\mathcal{Q}[p_{\boldsymbol{\theta}_m^{(t,s)}}(\boldsymbol{y}(\boldsymbol{x}_i) \,|\, \mathbf{y}, \mathcal{G}^{(t)})] = \{(y(\boldsymbol{x}_i)^c, w^c)\}_{c=1}^{C}$ and $\mathcal{Q}[q_{\boldsymbol{\theta}_m}(\boldsymbol{f}_m(\boldsymbol{x}_i))] = \{(\boldsymbol{f}_m(\boldsymbol{x}_i)^{c'}, w^{c'})\}_{c'=1}^{C'}$, as

$$\sum_{m=1}^{D_y} \Bigg\{ \sum_{i=1}^{N} \sum_{c=1}^{C} \sum_{c'=1}^{C'} w^c \; w^{c'} \log p_{\boldsymbol{\theta}_m}(y_m(\boldsymbol{x}_i)^c \,|\, \boldsymbol{y}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{x}_i)^c, \boldsymbol{f}_m(\boldsymbol{x}_i)^{c'}) \\ - \sum_{j=1}^{J_m} \mathrm{KL}(q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j}) \,||\, p_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j})) - \frac{J_m}{2} \left(\sum_{n \in \Pi_m^{\mathcal{G}}} |\boldsymbol{\lambda}_{m,n,j}| \right) \log N \Bigg\}. \quad (5.53)$$

As in the homogeneous model, optimization is performed using L-BFGS. For numerical stability, training alternates between optimizing $\boldsymbol{\theta}$ and optimizing $\boldsymbol{\mu}_m$ and $\boldsymbol{S}_m$.

5.10.6 Prediction in the Heterogeneous AcyGP Model

Just like in the homogeneous AcyGP model, training a heterogeneous AcyGP requires prediction on lines 3 and 4 of Algorithm 15, and also to produce samples used in query-driven adaptive sampling. Multivariate posterior predictions of the heterogeneous AcyGP model generally do not have closed form representations. However, we can either sample from the posterior distribution, or construct attribute quadrature rules.

Sampling from the Posterior Distribution

Sampling from the heterogeneous AcyGP posterior is performed using the posterior representation of the latent processes directly. For a vector of prediction inputs $\boldsymbol{X}^*$, distributions over the latent process posteriors $q_{\boldsymbol{\theta}_m}(\mathbf{f}_m^*)$ may be computed using (5.51). Samples of $\mathbf{f}_m^*$ may be taken directly from the multivariate Gaussian distributions.

A sample of $\mathbf{y}_m^*$ is computed using samples of $\mathbf{f}_m^*$ and samples of $\mathbf{y}_{\Pi_m^{\mathcal{G}}}^*$ according to the structural equation model of the AcyGP. The attribute dimensions are considered in a topological order consistent with the DAG structure $\mathcal{G}$. Using samples of $\mathbf{y}_{\Pi_m^{\mathcal{G}}}^*$, $t_{\boldsymbol{\lambda}_{\boldsymbol{m},\boldsymbol{n},\boldsymbol{j}}}(y_n(\boldsymbol{x}_i^*))$ are computed for all $n \in \Pi_m^{\mathcal{G}}$ and combined with samples of $\mathbf{f}_m^*$ to give samples of $\boldsymbol{\alpha}_m(\boldsymbol{x}_i^*)$. Finally, samples of $y_m(\boldsymbol{x}_i^*)$ are taken from the distribution $p_{\boldsymbol{\beta}_m}(y_m(\boldsymbol{x}_i^*) \,|\, \boldsymbol{\alpha}_m(\boldsymbol{x}_i^*))$.

Predictions as Quadrature Rules

We can construct quadrature rules for all attributes in the AcyGP at a single location in a similar manner to sampling. Starting with a quadrature rule for an empty set of variables $\mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}_{\emptyset}(\boldsymbol{x}_i^*))] = \{(\emptyset, 1)\}$, the quadrature rule is constructed incrementally, adding each attribute in topological order consistent with the DAG structure $\mathcal{G}$ as described in Algorithm 16.

The loop on line 2 loops through each attribute m, and has available the quadrature rule $\mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*))]$ for all attributes that precede attribute m in a topological ordering. For each element $\boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*)^c$ of the quadrature rule over the preceding attributes we construct $\mathcal{Q}[q_{\boldsymbol{\theta}}(y_m(\boldsymbol{x}_i^*) \,|\, \boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*)^c)]$ on lines 6 through 16. This is done using samples from a quadrature rule of $\boldsymbol{y}_{T_n(\mathcal{G})}$ on line 5 and samples from a quadrature rule for $q_{\boldsymbol{\theta}_m}(\boldsymbol{f}_m(\boldsymbol{x}_i^*))$ on line 11 to generate samples for $\boldsymbol{\alpha}_m(\boldsymbol{x}_i^*)$ on line 13. This is then passed through $p_{\boldsymbol{\beta}_m}$ to generate $\mathcal{Q}[q_{\boldsymbol{\theta}}(y_m(\boldsymbol{x}_i^*) \,|\, \boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*)^c)]$ on line 16.

We then combine elements of the quadrature rule over $\boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*)$ with elements of the quadrature rules over $\boldsymbol{y}_m(\boldsymbol{x}_i^*) \,|\, \boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*)^c$ to produce elements of the quadrature rule $\mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}_{T_m(\mathcal{G})}(\boldsymbol{x}_i^*))]$ on line 18. Once the final index m has been considered, the final quadrature rule is returned.

Algorithm 16: AcyGP Quadrature Prediction

Input : Trained AcyGP parameters $\boldsymbol{\theta}$, DAG $\mathcal{G}$, prediction location $\boldsymbol{x}_i^*$, and observed data $\mathbf{y}$

Output: Predictive quadrature rule $\mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}(\boldsymbol{x}_i^*))]$

$\mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}_{T_m(\mathcal{G})}(\boldsymbol{x}_i^*))] \leftarrow \{(\emptyset, 1)\}$
for m in order of topological ordering $T(\mathcal{G})$
  $\mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*))] \leftarrow \mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}_{T_m(\mathcal{G})}(\boldsymbol{x}_i^*))]$
  $\mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}_{T_m(\mathcal{G})}(\boldsymbol{x}_i^*))] \leftarrow \{\}$
  for $(\boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*)^c, w^c) \in \mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*))]$
    if $y_m(\boldsymbol{x}_i^*)$ is observed **then**
      $\mathcal{Q}[q_{\boldsymbol{\theta}}(y_m(\boldsymbol{x}_i^*) \,|\, \boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*)^c)] \leftarrow \{(y_m(\boldsymbol{x}_i^*), 1)\}$
    else
      Construct $\mathcal{Q}[q_{\boldsymbol{\theta}_m}(\boldsymbol{f}_m(\boldsymbol{x}_i^*))]$ using Gauss-Hermite quadrature on the distribution constructed in (5.51)
      $\mathcal{Q}[q_{\boldsymbol{\theta}}(y_m(\boldsymbol{x}_i^*) \,|\, \boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*))] \leftarrow \{\}$
      for $(\boldsymbol{f}_m(\boldsymbol{x}_i^*)^{c'}, w^{c'}) \in \mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{f}_m(\boldsymbol{x}_i^*)]$
        for $j \in \{1, \ldots, J_m\}$
          $\alpha_{m,j}(\boldsymbol{x}_i^*)^{c'} \leftarrow f_{m,j}(\boldsymbol{x}_i^*)^{c'} + \sum_{n \in \Pi_m^{\mathcal{G}}} t_{\boldsymbol{\lambda}_{m,n,j}}(y_n(\boldsymbol{x}_i^*)^c)$
        Construct $\mathcal{Q}[p_{\boldsymbol{\beta}_m}(y_m(\boldsymbol{x}_i^*) \mid \boldsymbol{\alpha}_m(\boldsymbol{x}_i^*))]$ using an appropriate quadrature rule for $p_{\boldsymbol{\beta}_m}$
        for $(y_m(\boldsymbol{x}_i^*)^{c''}, w^{c''}) \in \mathcal{Q}[p_{\boldsymbol{\beta}_m}(y_m(\boldsymbol{x}_i^*) \mid \boldsymbol{\alpha}_m(\boldsymbol{x}_i^*))]$
          Add $(y_m(\boldsymbol{x}_i^*)^{c''}, w^{c'} \times w^{c''})$ to $\mathcal{Q}[q_{\boldsymbol{\theta}}(y_m(\boldsymbol{x}_i^*) \,|\, \boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*))]$
    for $(y_m(\boldsymbol{x}_i^*)^{c'}, w^{c'}) \in \mathcal{Q}[q_{\boldsymbol{\theta}}(y_m(\boldsymbol{x}_i^*) \,|\, \boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*)^c)]$
      Add $([\boldsymbol{y}_{T_n(\mathcal{G})}(\boldsymbol{x}_i^*)^{c'}, y_m(\boldsymbol{x}_i^*)^c], w^c \times w^{c'})$ to $\mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}_{T_m(\mathcal{G})}(\boldsymbol{x}_i^*))]$
  if m is last index in $T(\mathcal{G})$ **then**
    return $\mathcal{Q}[q_{\boldsymbol{\theta}}(\boldsymbol{y}_{T_m(\mathcal{G})}(\boldsymbol{x}_i^*))]$

5.11 Experiments

We now present experimental results of the AcyGP model performed on synthetic and real data. We hypothesize that the the AcyGP model will be able to produce more accurate predictions of missing data than other Gaussian process models in prediction tasks with strong correlations between attributes. We show improvements in prediction error measured from the predictive mean of a Gaussian process over state of the art Gaussian process methods that correlate all attributes, and a model that is structures as an AcyGP, but maximizes unpenalized data likelihood.

Our synthetic experiment is designed to demonstrate how correlating independent attributes can leading to highly incorrect predictions. The real data experiments use common MOGP benchmarks, but also demonstrate the applicability of the AcyGP model in multiple, dissimilar domains both relevant to and independent from adaptive sampling.

We tested the homogeneous AcyGP model against the frequently applied MOGP baseline SLFM, and the recent state of the art GPAR [110] on continuous valued data. We tested against the heterogeneous Gaussian process model [96], which models heterogeneous distribution parameters through a fully correlated SLFM model, on binary-valued data. The experiments were each repeated 10 times, with random initial hyperparameters used for optimization, and no edges in the initially assumed DAG structure $\mathcal{G}^{(0)}$.

When a single attribute is to be predicted, we present results in terms of mean absolute error, which is the mean error of the difference between the predicted mean and the true data point. When multiple attributes are predicted, we follow the literature and present errors for each attribute in terms of standardized mean squared error (SMSE), defined below, where y_{pred} is predicted by the GP, y_{true} is the true value of the removed test data, and μ_{test} is the mean of the removed test data for the attribute.

$$\text{SMSE} = \frac{\sum_{i=1}^{N_{pred}} (\mathbb{E}[y_{pred}(\boldsymbol{x}_i)] - y_{true}(\boldsymbol{x}_i))^2}{\sum_{i=1}^{N_{pred}} (\mu_{test} - y_{true}(\boldsymbol{x}_i))^2} \tag{5.54}$$

5.11.1 Synthetic Data Experiment

This experiment was formulated to test the capability of the AcyGP model to recover known structure and avoid spurious correlations that would harm prediction accuracy. To do so, we generate synthetic data with attributes that are strongly correlated, but independent when conditioned on a third attribute.

Our synthetic data consisted of 41 equally spaced data points on the domain $[0, 10]$ for the following functions with no noise.

$$y_1(x) = e^{-(x-2)^2} \quad y_2(x) = 1.1e^{-x^2} + 5e^{-0.4(x-2.5)^2} + 3e^{-4(x-7)^2} + 0.01x^3$$
$$y_3(x) = 2y_1(x) + 0.8\log(x + 0.25)$$

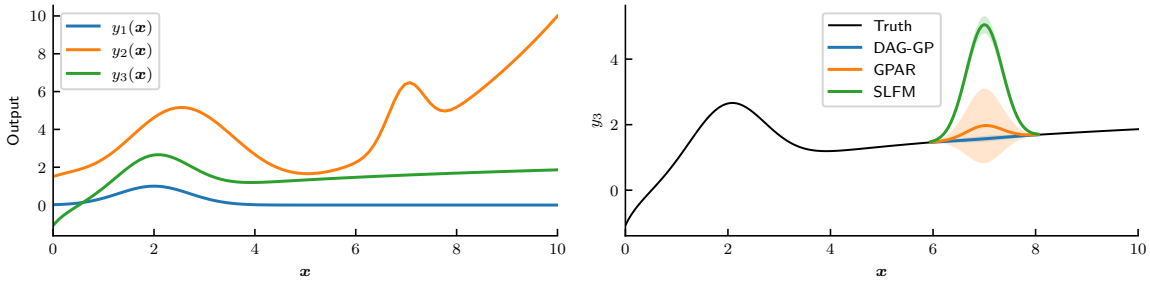
The functions are shown graphically in Figure 5-6a. We then remove and attempt to predict y_3 in the range $[6, 8]$ using an AcyGP with RBF kernels.

The functions in this experiment are designed so that y_3 is strongly correlated with y_1, and weakly correlated with y_2 through peaks near $x = 2$. However, y_3 depends only on y_1, and y_2 has a unique peak at $x = 7$. If an MOGP method falsely correlates y_3 with y_2, we expect a peak to appear in the prediction of y_3.

Unlike SLFM and GPAR, the AcyGP model does not introduce spurious correlations between y_2 and y_3, improving prediction accuracy. Structural search reveals that while y_3 is well described by dependence on either y_1 or y_2, dependence on y_2 contributes little additional explanatory power when y_1 is already included as a parent, so there is limited evidence for a second peak in y_3. The AcyGP model performs this tradeoff and recovers the true DAG, with a single edge $y_1 \to y_3$, as shown in Figure 5-7. A second peak is not predicted, as in Figure 5-6b. Mean absolute errors of the predicted means are given in Table 5.2, showing small errors for the AcyGP model. GPAR and SLFM directly correlate y_2 with y_3, resulting in a second peak in predictions of y_3 and significantly larger mean absolute errors.

Table 5.2: Mean absolute errors of predictions on the synthetic data set.

SLFM	GPAR	AcyGP
1.00 ± 0.12	0.151 ± 0.004	**0.0086 ± 0.0003**



(a) Functions ground truth.

(b) Predictions of y_3.

Figure 5-6: Synthetic data set visualization and predictions with 95% confidence intervals.

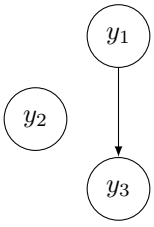


Figure 5-7: Structure extracted by the AcyGP model on the synthetic data set.

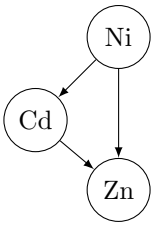


Figure 5-8: Structure extracted by the AcyGP model on the Jura data set.

5.11.2 Jura Data Set

The Jura data set [54] consists of soil metal concentrations collected at different spatial locations in the Swiss Jura. This dataset is similar to the type of observations found in typical adaptive sampling problems, as it contains geological variables distributed through space with varying longitude and latitude. For this reason, the Jura dataset is a common benchmark for geospatial modeling.

We repeat the experimental setup in other MOGP studies [4, 54, 110], where 259 observations of nickel (Ni), zinc (Zn), and cadmium (Cd) concentrations are known. 100 additional data points provide observations of Ni and Zn, and Cd is predicted at those 100 locations. As in previous works, the data is log-transformed then modeled using RBF kernels, though errors are computed in the original scale.

The structure extracted by the AcyGP model is given in Figure 5-8. The DAG is fully connected, which reflects the fact that all metal concentrations are well correlated in this experiment. But this experiment shows that the AcyGP model has value even with strongly correlated variables, by reducing mean absolute error over the state of the art. It achieves this by solving for an ordering of variables that better matches the relationships seen in the data. Figure 5-8 implies that Zn is modeled as linearly dependent on both Cd and Ni, so that additional information about concentrations of Cd can be extracted from the spatial distribution of concentrations of Cd.

Mean absolute errors of the cadmium predictions are given in Table 5.3. Despite using simpler expressions for inter-attribute relationships than the previous best-known method, GPAR, the AcyGP model achieves a new state of the art in minimization of mean absolute error on this baseline. This improvement results directly from the solved structural model, where Cd is not modeled as a function of

Table 5.3: Mean absolute errors on Jura prediction task.

SLFM	GPAR	AcyGP
$0.5352 \pm 1\times10^{-5}$	$0.3999 \pm 4\times10^{-4}$	**0.3946 ± 0.023**

function of both Ni and Zn, as is assumed in GPAR. Instead, the solved DAG models Cd as child of Ni, and Zn as a child of Ni and Cd.

5.11.3 Exchange Data Set

The exchange data set consists of the daily exchange rates with respect to USD of precious metals and currencies over 251 days of 2007. This domain is frequently tested in the MOGP literature, and is an insightful test of the AcyGP model because there is a large number of attributes with complex interdependencies and temporal correlations over many length scales. Testing on this domain also shows the applicability of the AcyGP model outside of adaptive sampling experiments.

Previous works on this data set assumed that global markets are driven by a small number of latent forces and are highly correlated [98]. We instead hypothesize that currencies are driven by a few select trading partners, and test whether a sparse DAG structure improves prediction accuracy. We model the prices of gold (AUX), silver (AGX), and 6 currencies (CAD, EUR, CHF, AUD, NZD, and HKD) using rational quadratic kernels. The task is to predict CAD on data points 50-100, HKD on points 100-150, and AUD on points 150-200, given all other observations. We consider a subset of possible DAG structures where the only edges possible are from elements of {AUX, AGX, NZD, EUR, CHF} to elements of {CAD, HKD, AUD}. No restrictions are placed on other methods. To test the influence of sparsity in the DAG, we repeat the experiment with all possible edges from {AUX, AGX, NZD, EUR, CHF} to {CAD, HKD, AUD}, labeled as 'AcyGP Full'.

Standardized mean squared errors (SMSEs) are given in Table 5.4. Predictions for CAD and HKD are given in Figures 5-9a and 5-9b, and the learned DAG is given in Figure 5-9c. Compared to alternate methods, CAD predictions in the AcyGP model are lower and closer to truth. Additionally, confidence intervals in the predictions

Table 5.4: Standardized mean squared error on the exchange prediction task.

	SLFM	GPAR	AcyGP	AcyGP Full
CAD	1.63 ± 0.10	0.953 ± 0.022	$\mathbf{0.6010 \pm 1 \times 10^{-8}}$	$0.6425 \pm 2 \times 10^{-4}$
HKD	1.15 ± 0.11	1.380 ± 0.075	$\mathbf{0.6353 \pm 6 \times 10^{-8}}$	1.75 ± 0.15
AUD	0.099 ± 0.033	0.04091 ± 0.00095	$\mathbf{0.02920 \pm 2.2 \times 10^{-4}}$	$0.02936 \pm 1 \times 10^{-6}$
Average	0.959 ± 0.082	0.791 ± 0.028	$\mathbf{0.4218 \pm 7 \times 10^{-5}}$	0.807 ± 0.050

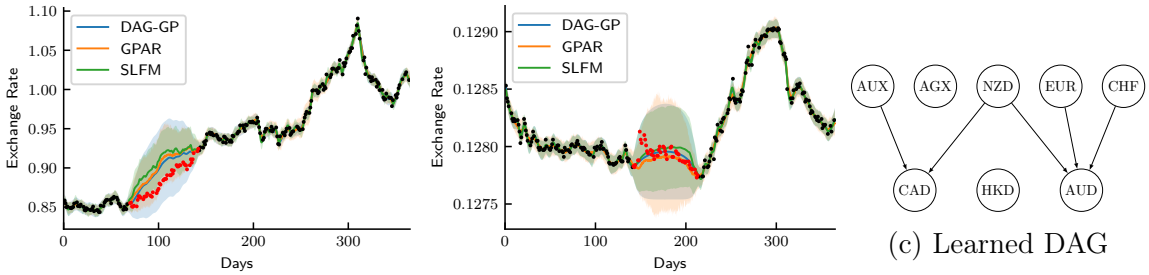
are substantially larger in the AcyGP model. Typically, larger confidence intervals are undesirable for prediction, but in this experiment, larger uncertainties better model the deviation of the data from the predicted mean. Some data is outside the 95% confidence interval generated by the SLFM and GPAR models, resulting in low likelihood, and giving greater likelihood of the predictions in the AcyGP model despite the larger variance in predictions. This shows that correlating all attributes will tend to lead to overestimates of confidence, which may not reflect certainty about the data.

HKD is identified to have no parents due to its unique slump over the first half of the year. This corresponds to meaningful domain-specific structure, since the price of HKD is known to be tied to USD [79], and it does not respond to market forces that affect other currencies relative to USD. The predicted price of HKD is pushed upwards by the SLFM and downwards by GPAR, but predicting this attribute with no parents (as an independent GP) leads to more accurate predictions.

Using a fully connected structure leads to significantly reduced prediction accuracy, particularly for HKD. The fully connected model learns correlations between unrelated currencies, introducing patterns into the prediction targets that do not occur. The AcyGP model avoids introducing these patterns, and so improves prediction accuracy.

5.11.4 Andromeda Data Set

The Andromeda data set [58, 128] consists of daily averages of temperature (T), pH, conductivity (σ), salinity (S), oxygen (O_2), and turbidity (β) observed over 54 days



(a) CAD predictions (b) HKD predictions (c) Learned DAG

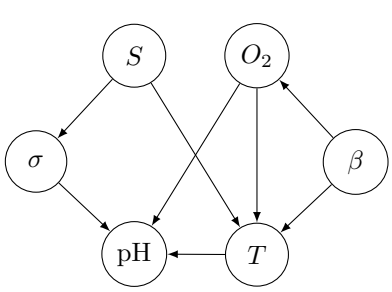
Figure 5-9: Exchange test results. Predictions show 95% confidence intervals, with missing data in red.

Table 5.5: Standardized mean squared error on the Andromeda prediction task.

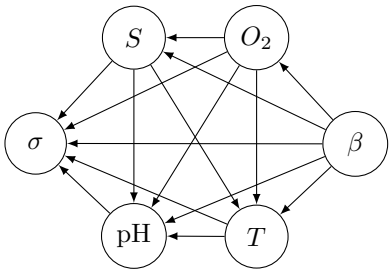
	SLFM	GPAR	AcyGP	AcyGP Full
Salinity	0.0934 ± 0.0057	0.2087 ± 0.0024	$\mathbf{0.05324 \pm 1 \times 10^{-8}}$	0.0962 ± 0.0020
Oxygen	0.130 ± 0.019	0.4626 ± 0.0027	$\mathbf{0.03210 \pm 5 \times 10^{-9}}$	0.0387 ± 0.0021
Average	0.112 ± 0.011	0.3356 ± 0.0020	$\mathbf{0.04267 \pm 8 \times 10^{-9}}$	0.0675 ± 0.0021

by an underwater mooring. Oceanographic models causally describe many of these variables as functions of small sets of others, so we anticipate a sparse DAG model to well describe this system. Furthermore, since the number of data points is small, we expect spurious statistical correlations to exist in this data set. We tested the capability to predict salinity on days 21-30 and oxygen on days 31-40 given all other data. We also repeated the test, solving for the highest unpenalized score AcyGP ('AcyGP Full') for reference.

Standardized mean squared errors are given in Table 5.5. We see significant improvement in prediction under the AcyGP model. The optimized DAG, as shown in Figure 5-10a places salinity and oxygen near the top of the DAG, with 2 and 3 direct relatives. The learned DAG structure appears to avoid modeling additional correlations that harmfully influence predictions in the other, fully correlated models. All edges to and from oxygen and salinity in the sparse structure appear in the fully connected structure learned without penalization in Figure 5-10b, so the presence of the additional direct correlations to other variables is responsible for reduced prediction accuracy by the fully connected AcyGP model.



(a) AcyGP learned DAG



(b) AcyGP learned DAG without edge penalization

Figure 5-10: Learned structures in the Andromeda prediction task.

5.11.5 Seep Data Set

We construct a data set describing likely sites of natural hydrocarbon seepage in the Costa Rica continental margin, derived from observations taken by Sahling et al. [114]. The adaptive sampling technology described in this thesis was applied on a field deployment to this continental margin in December 2018 [9, 144], so testing on this data set is an accurate representation of the type of environmental prediction required for adaptive sampling missions.

The seep data set describes longitude and latitude positions of 112 candidate seep sites. A location is considered to be a candidate seep site when a visual or chemical confirmation of seepage has been detected, or an observable signal strongly correlated with seepage is present, such as specific bathymetric features, microbial mats, or backscatter signals. For each candidate seep site, our data set includes presence of a mound (M), a pockmark (P), a fault (F), elevated backscatter (B), and confirmed presence of seepage (S), each of which are binary variables.

We model prediction of the presence of seepage using the heterogeneous AcyGP model and heterogeneous Gaussian processes [96] with RBF kernels. In this experiment, the heterogeneous AcyGP model is used with a fixed structure, shown in Figure 5-11. This model is derived from a simplification of a geologists intuitive model of an environment. The occurrence of bathymetric features and backscatter is considered to be an independent event, unless connected through seepage as a common effect.

In each experiment, we remove seep presence data from 8 locations for training,

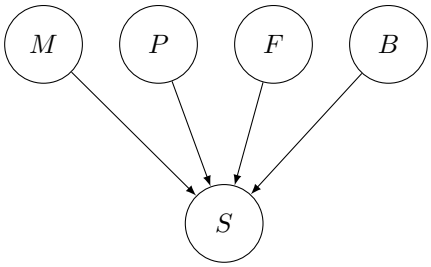


Figure 5-11: Imposed AcyGP structure in the seep data set experiment.

Table 5.6: Mean absolute errors on the seep prediction task.

Heterogeneous GP	Heterogeneous AcyGP
0.303 ± 0.044	$\mathbf{0.066 \pm 0.035}$

and record the mean error of the probability of seep presence at the locations where it was removed. Both models are variational, and we use all 112 inputs as the latent locations. Errors are reported with respect to 6 repeats of this experiment, with different data withheld on each repeat. For a given repeat, the withheld data is the same for both models.

Mean absolute errors from the experiment are shown in Table 5.6. The heterogeneous AcyGP model has only 21.8% of the mean absolute error of the heterogeneous Gaussian process model in this experiment. Examination of the optimized AcyGP reveals that the parameters $\boldsymbol{\lambda}_{m,n,j}$ which connect probability of seepage to its parents, are very large in magnitude. This implies that most of the information allowing prediction of seepage comes from other observations at the same spatial location, rather than spatial correlations. The AcyGP model with a binary attribute y_m and a binary parent y_n assigns two parent dependence parameters in the vector $\boldsymbol{\lambda}_{m,n,j}$, one for when the parent $y_n(\boldsymbol{x}_i) = 1$, and another when $y_n(\boldsymbol{x}_i) = 0$. This makes the AcyGP model more expressive than the heterogeneous GP model, which uses only one parameter per attribute per latent process. The AcyGP model is better able to capture the true distribution of seepage given observable features, resulting in lower error predictions.

5.12 Summary

In this chapter, we introduced the AcyGP model. The AcyGP model aims to improve modeling accuracy in data limited environments, where spurious correlations arise from small-sample statistics. The AcyGP model constructs an environment from multiple Gaussian processes that are combined in a directed acyclic graph model structure. The DAG structure allows an expert to specify known relationships between attributes that are known to exist and known not to exist, and the remainder are learned from data using score-based structure learning. When structure is learned, it is limited to edges which are strongly justified, preventing spurious correlations from being introduced into the model. The formulation of an AcyGP also makes parameterizations of relationships between attributes easily interpretable, so that qualitative expert knowledge of monotonicity and relative scale can be encoded as constraints and extracted from trained models.

In addition, we introduced the homogeneous AcyGP model, where all attributes are continuous and unbounded, and the more general heterogeneous AcyGP model, which allows categorical and bounded-domain attributes. We then showed experimentally that the sparse models generated by the AcyGP model outperform existing methods. On a diverse set of benchmarks where sparse models are solved, the AcyGP model achieves errors 38.1% as large or less compared to other Gaussian process approaches. In domains where a fully connected model is solved, the error improvements are more marginal, but the AcyGP model still achieves state of the art performance.

Chapter 6

Optimizing Structure Using A* with Bounding Conflicts

In the last chapter, we argued for the importance of searching for the best model structure, as well as mode parameters, given a expert's prior knowledge. But training a complex machine learning model like an AcyGP is highly time consuming, and when the best DAG structure is selected by training each attribute on all possible parents, learning progresses slowly. The space of possible models is large, so search must be efficient and fast to prevent unnecessary slowdown during adaptive sampling. In this chapter we accelerate structure learning by drawing upon an approach to speed up learning, called bounding conflicts, taken from the model-based reasoning community, for performing system-wide estimation. The key insight behind bounding conflicts is to produce a more informed generator of candidate structures, by learning from the results of performing detailed parameter estimation on previous structures.

In this chapter, we describe the use of A* with bounding conflicts (A*BC) for finding maximum penalized likelihood structural models when those likelihoods are expensive to compute. Our primary motivation is rapidly solving for optimal directed acyclic graph structures in the AcyGP model that are consistent with expert imposed constraints. Use of bounding conflicts allows optimal structures to be found without explicitly evaluating the likelihoods of all models.

Structure learning is typically applied in problems where likelihood of any struc-

ture can be easily computed. A typical step before searching over the space of structures is to generate the maximum likelihood of each variable conditioned on all possible parent sets [156, 33]. When these likelihoods are difficult to compute, this preprocessing step takes most of the time used for structure learning. Our idea is to compute the likelihoods during the course of search over structures, and use what is learned from candidate structures to limit further evaluation to fewer structures. Our particular approach is based on the A*BC algorithm [136], which was introduced to incrementally improve a heuristic from evaluated candidates within hybrid estimation problems.

Our algorithm introduces several innovations to A*BC. We provide a method of bounding likelihood that is applicable for structure learning, which is derived from evaluated likelihoods of models with more edges. This makes it advantageous to compute likelihoods for models with many edges first, and we introduce a method to selecting likelihoods to compute that are expected to be tight bounds on likelihoods of unevaluated models. We further reduce the number of models that must be evaluated by placing bounds on the cost of paths already traversed in A* search, in addition to bounds on the cost to go, as has previously been done in A*BC. In total, our approach reduces the number of models evaluated significantly, with a large reduction in solution time. We show that in AcyGP structural optimization, A*BC performs approximately 50% faster than other exact structural search algorithms with no loss in optimality, and is comparable in time to a well-tuned local search.

6.1 Motivation

Environments of interest for adaptive sampling, such as those in the oceans or on other planetary bodies, consist of many interacting variables of potential interest. These environments are frequently under study because the nature of the interactions is unknown. Yet even when the environment is well understood, a complete environmental model may be outside of the expertise of the team conducting the experiment, or depend on unobservable parameters. For example, sea floor hydrocarbon release

is strongly influenced by subsurface geology, which is only partially observable, and an expert may consider multiple models to be consistent with observations of the environment. Our environment modeling approach must still be able to be used to perform inference in the face of this model uncertainty.

To resolve model uncertainty, we learn the unspecified part of the structure from the data. For the AcyGP model, this is performed by optimizing BIC, with the objective described in Chapter 5. However, searching over the space of DAG structures is highly time consuming, particularly when considering data correlated across inputs. The difficulty arises from the time needed to evaluate BIC, which requires training a Gaussian Process in order to evaluate the optimized likelihood of the data. Almost all time in structural search is spent evaluating optimized likelihoods, so to see significant decreases in search time, the number possible structures that is trained must be reduced.

We present a novel variant of A* with bounding conflicts [136] that substantially accelerates finding the optimal structural model. The algorithm uses a set of optimized likelihoods to place upper bounds on the likelihoods of other parent sets. The bounds allow the optimal structure to be found while computing the optimized likelihood of only a fraction of possible structures.

6.2 Overview of Structural Search Using A* with Bounding Conflicts

In this section, we give an intuitive overview of the idea behind structure learning using A* with bounding conflicts. We wish to maximize the penalized likelihood of a DAG, say using BIC, and we will do so without evaluating the likelihood of every possible parent set. The key idea is that evaluated likelihoods can be used to place bounds on other, unevaluated likelihoods, and this allows us to prove structure optimality without evaluating every single likelihood.

The essence of this idea can be seen from the example DAG given in Figure 6-

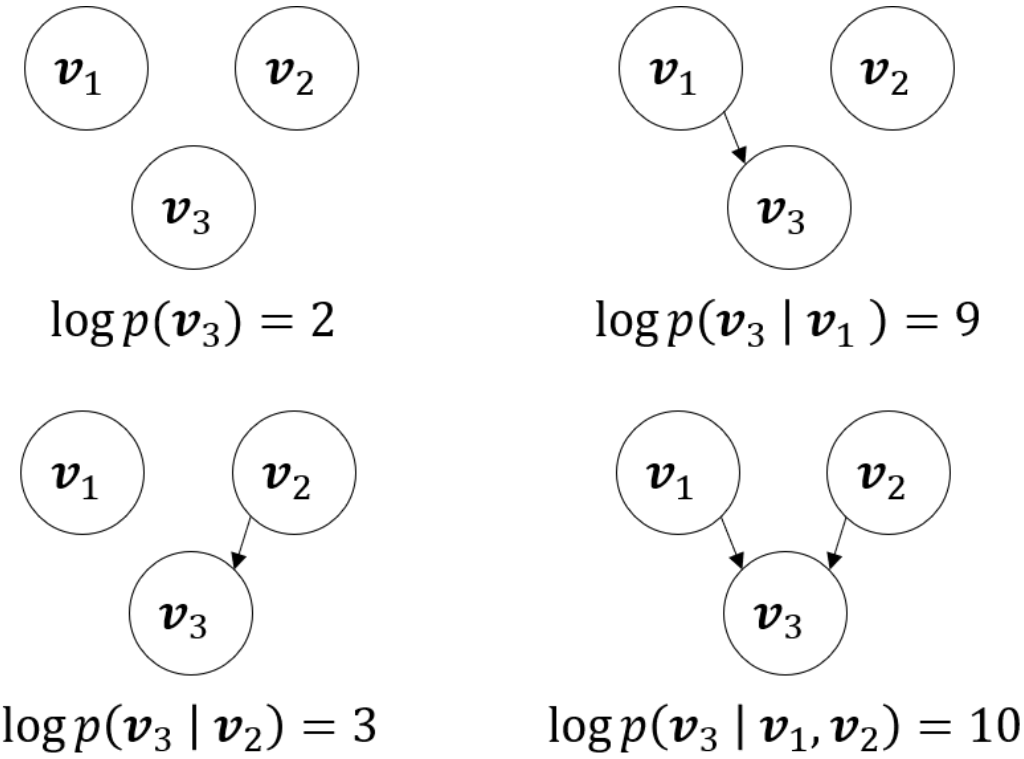


Figure 6-1: Example optimized log likelihoods for variable $\mathbf{v}_3$ conditioned on different sets of parents. If we wish to maximize a sum of optimized log likelihood and an edge penalization of 2 per edge, it is never necessary to evaluate the likelihood of $\mathbf{v}_3$ with no parents.

1. To avoid polluting the example with too many numbers, let us assume that we already know $\mathbf{v}_1$ and $\mathbf{v}_2$ have no parents, and we need to find the parents of $\mathbf{v}_3$. In analogy to maximizing BIC, we wish to find the parents with the largest optimized log likelihood, minus a penalization of 2 per edge in the model. In this case, the answer is the structure in the top right, with a penalized likelihood of 7.

One option is to evaluate every likelihood in Figure 6-1 and pick the largest penalized likelihood, but then we have to evaluate all four likelihoods. Instead, we can use the fact that adding edges must increase maximum likelihood, and show that the top left structure never needs to be evaluated.

Let us assume we evaluated the optimized likelihood for all structures except for $\mathbf{v}_3$ with no parents. Intuitively, a DAG structure with an edge from $\mathbf{v}_2$ to $\mathbf{v}_3$ can still represent any probability distribution that matches a DAG with no edges to $\mathbf{v}_3$, so

$\max \log p(\mathbf{v}_3 \mid \mathbf{v}_2) \geq \max \log p(\mathbf{v}_3)$. The maximum penalized likelihood for the top left structure is therefore less than or equal to $\max \log p(\mathbf{v}_3 \mid \mathbf{v}_2) - 2 = 1$. But we already know that $\mathbf{v}_1$ with $\mathbf{v}_3$ as a parent achieves penalized likelihood of 7, so no parents for $\mathbf{v}_3$ cannot be optimal, and we don't have to determine the likelihood.

While useful, this bound doesn't tell us which order to evaluate the likelihoods. If we did not know $p(\mathbf{v}_3 \mid \mathbf{v}_2)$, we would not know that $p(\mathbf{v}_3)$ cannot be optimal. Our solution is to use A* search with bounding conflicts for this purpose. Using the known bounds, A* search allows us to use heuristic information to select the structures that are believed to be optimal under current knowledge of likelihoods. A* with bounding conflicts is used to improve the quality of the heuristic as search progresses and additional information about likelihoods is gathered.

In Section 6.3, we review related work in structure learning and A* search. In Section 6.4 we present a formal problem statement, then in Section 6.5 we review key concepts behind A* and A* with bounding conflicts. Our approach is described technically in Section 6.6. We prove the optimality of the approach in Section 6.7, show how to use expanded lists in Section 6.8, and compare A* to A* with bounding conflicts in Section 6.9. Finally, we test our approach, and show speedups compared to A* search and local searches using in Section 6.11.

6.3 Related Work

In this section we review work related to our structure learning approach. We will build off existing structure learning algorithms, while using utilizing bound-directed search developed in the conflicted-directed search community.

6.3.1 Structure Learning Methods

In this thesis we build upon score-based methods of structure learning. In this section we review score-based structure learning, and contrast it to constraint-based structure learning. We also discuss the distinction between exact and local search in structure learning.

Algorithms that learn directed acyclic graphs from data are typically classified as either constraint based methods or score based methods. Score based structure learning algorithms optimize a score function that rewards high likelihood of the observed data and penalizes a large number of edges in the DAG. Exact score based algorithms, which identify globally optimal structures, perform well in the presence of limited data, but become intractable to optimize for large numbers of variables, because the number of possible DAGs grows super-exponentially with the number of variables considered [111]. State of the art exact score based methods include A* [156] and branch and bound [37] based search, which generate bounds on score by loosening acyclicity constraints during search. Alternative approaches have encoded the problem as an integer linear program, made more efficient through heuristics that greedily construct high scoring solutions [13, 33, 104].

Typically, DAG structure learning algorithms have been applied to problems where the score function to optimize is relatively inexpensive to compute, so that the difficulty in optimization comes from the size of the DAG search space. In contrast, the DAGs we consider have relatively few variables, typically less than 20, so structural search is fast, but most complexity arises from evaluating the scores. The previously mentioned score-based algorithms do not attempt to limit the number of scores evaluated, and most require a full list of scores as input, making them unsuitable for our purposes. Our approach is able to substantially reduce overall optimization time by limiting the number of scores that must be evaluated during search.

Constraint based methods estimate conditional independence relations from the data, and then generates candidate DAGs consistent with those conditional independence relations [106]. Constraint based methods tend to scale well to large numbers of variables, but may be inaccurate in the presence of limited data, where conditional independence tests may be inaccurate. Modern constraint based methods have focused on limiting the number of independence tests necessary. For example, the PC algorithm [127] tests conditional independence in progressively larger sets, while other approaches construct a candidate DAG using independence tests on small variable sets and attempt to repair detected inconsistencies [28, 38]. A separate approach

instead learns smaller DAGs that are later combined [153]. It is more straightforward to generate bounds on likelihood from evaluated likelihoods than it is to generate bounds on conditional independence tests. For this reason, we take a score-based learning approach.

Local score based algorithms instead search the space of DAG structures locally. Search is defined on the space of DAGs, DAG Markov equivalence classes [30], or variable orderings [133], with a second level of search defined in the latter cases where more than one DAG is consistent with each element of the search space. Search space elements are mutated to define a neighborhood of structures, and a hill climbing strategy is followed to reach a locally optimal structure. Stochastic search methods are employed to escape local optima, such as Tabu search [52, 121], which prevents previously evaluated DAGs from being revisited, and simulated annealing [82], which allows transitions to suboptimal structures with a probability based on their score. Since the full space of possible DAGs is not considered in local methods, fewer score evaluations are necessary. However, convergence to the true optimal structure is not guaranteed, and there are no methods to evaluate the degree of suboptimality of the DAG returned, preventing it from being interpreted meaningfully. Where speed is favored over optimality, AcyGP structures can be optimized with local search methods. However, we show experimentally that A*BC search reaches provably optimal structures in time consistent with a well-tuned local search algorithm. This is significant, because the tuned parameters would not be known for most problems, and A*BC does not require these parameters.

6.3.2 A* and Conflict-Directed Search

In this chapter, we use this idea of bound generation during search that was developed for A* with bounding conflicts, and apply it in a structure learning setting. A* was originally proposed by Hart et al. [57] for motion planning problems. A* expands may be formulated as search over a graph of states, where the objective is to find a path of maximum reward or minimum cost from a start vertex to a goal vertex. A* does this through the use of a heuristic, which provides a bound on the cost of

the optimal path. At each step of the algorithm, the next best edge in the graph, evaluated using the heuristic, is selected and explored.

While A* has been adopted in a number of domains and is still widely used [6, 137], it does not improve its heuristic as information is gained. Conflict-directed A* [146] improved upon A* by using information about inconsistency with constraints. A* is used to select states consisting of variable assignments in order of maximum likelihood, while rules for inconsistency with observations are derived from assignments. These rules are generalized to prune additional states from the search space. A* with bounding conflicts [136] further generalizes this idea, by deriving stricter bounds on the likelihoods of variable assignments as they are considered during search. These bounds are used to inform more precise heuristics, reducing the number of states that need to be considered searched.

6.4 Problem Statement

We are motivated by the problem of finding a directed acyclic graph that maximizes a score function. The score is separable between parent sets and composed of an expensive to optimize likelihood function and a complexity penalty. Partial expert knowledge of structure is encoded through sets of allowable parents for each variable, which restricts the search space.

Let us set up this problem from the perspective of Problem 3 in Chapter 5, where we were interested in finding the structure that optimized the BIC for observations in an AcyGP, subject to restrictions on the allowable parent sets in the DAG structure. In order to solve Problem 3, we needed to find the structure that optimized a more complex objective function within structural EM. In the heterogeneous AcyGP case,

we sought the parameters and structure that satisfied the following complex function

$$\arg\max_{\boldsymbol{\theta},\mathcal{G}} \sum_{m=1}^{D_y} \mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}} \left[\sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\theta}_m}(\boldsymbol{f}_m(\boldsymbol{x}_i))} \left[\log p_{\boldsymbol{\theta}_m}(y_m(\boldsymbol{x}_i) \,|\, \boldsymbol{y}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{x}_i), \boldsymbol{f}_m(\boldsymbol{x}_i)) \right] \right] - \sum_{j=1}^{J_m} \mathrm{KL}(q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j}) \,||\, p_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j})) - \frac{J_m}{2} \left(\sum_{n \in \Pi_m^{\mathcal{G}}} |\boldsymbol{\lambda}_{m,n,j}| \right) \log N. \quad (6.1)$$

To avoid the complexity associated with this expression, and to generalize the content of this chapter to other problems, we present this problem as one of finding the parameters and structure that satisfy

$$\arg\max_{\boldsymbol{\theta},\mathcal{G}} \sum_{m} \rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) - c_m(\Pi_m^{\mathcal{G}}), \quad (6.2)$$

for some function of variables and parent sets $\rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}})$ that depends upon parameters $\boldsymbol{\theta}_m$, and a DAG complexity penalization $c_m(\Pi_m^{\mathcal{G}})$. The problem of finding the AcyGP structure with maximum BIC is recovered using each $\mathbf{v}_m$ as the vector of environment variables of output m, and the functions

$$\rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) = \mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}} \left[\sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\theta}_m}(\boldsymbol{f}_m(\boldsymbol{x}_i))} \left[\log p_{\boldsymbol{\theta}_m}(y_m(\boldsymbol{x}_i) \,|\, \boldsymbol{y}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{x}_i), \boldsymbol{f}_m(\boldsymbol{x}_i)) \right] \right] - \sum_{j=1}^{J_m} \mathrm{KL}(q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j}) \,||\, p_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j})). \quad (6.3)$$

and

$$c_m(\Pi_m^{\mathcal{G}}) = \frac{J_m}{2} \left(\sum_{n \in \Pi_m^{\mathcal{G}}} |\boldsymbol{\lambda}_{m,n,j}| \right) \log N, \quad (6.4)$$

as given by (5.52).

The generalized DAG structure learning problem is formalized in Problem 4 below.

Problem 4. *Consider of a set of variables* $\mathcal{V} = \{\mathbf{v}_m\}_{m=1}^{D_y}$. *For each variable* $\mathbf{v}_m \in \mathcal{V}$ *and set* $\mathbf{v}_\Psi \subseteq \mathcal{V} \setminus \{\mathbf{v}_m\}$, *suppose there exists a function* $\rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_\Psi)$ *with output in* $\mathbb{R}$ *that is parameterized by the vector* $\boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Psi}$. *Let* $\mathcal{P}_{m,\Psi} = \{\rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_\Psi) \,|\, \boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Psi}\}$ *be the set of function outputs generated by all possible parameterizations, and*

let $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi) = \max_{\boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Psi}} \rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_\Psi)$. *Assume that the functions* $\rho_{\boldsymbol{\theta}_m}$ *satisfy* $\mathcal{P}_{m,\Psi} \subseteq \mathcal{P}_{m,\Phi}$ *for all* m *and* $\Psi \subseteq \Phi$.

Given a score function $Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^\mathcal{G}}) = \hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^\mathcal{G}}) - c_m(\Pi_m^\mathcal{G})$ *with cost functions* $c_m : 2^{[D_y]\setminus\{m\}} \to \mathbb{R}$, *and allowable parent sets* $\{\Xi_m\}_{m=1}^{D_y}$, *find the directed acyclic graph* $\mathcal{G}^* = (\mathcal{V}, \mathcal{E})$ *and vector of parameters* $\boldsymbol{\theta}$ *that satisfies*

$$\begin{aligned} \arg\max_{\boldsymbol{\theta},\mathcal{G}} \quad & \textstyle\sum_{m=1}^{D_y} Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^\mathcal{G}}) \\ \text{such that} \quad & \Pi_m^\mathcal{G} \in \Xi_m \\ & \hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^\mathcal{G}}) \text{ expensive to compute} \end{aligned}$$

The condition that $\mathcal{P}_{m,\Psi} \subseteq \mathcal{P}_{m,\Phi}$ means that the effects of edges can be 'zeroed out', so that a DAG with a larger set of edges represents a larger set of distributions. This is necessary so that bounds on likelihood apply.

The restriction that $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^\mathcal{G}})$ is expensive to compute is not a hard restriction; the methods described in this chapter can be applied regardless. However, when $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^\mathcal{G}})$ is easily computed, existing DAG structure learning algorithms are typically substantially faster than our approach. Though we do not precisely characterize the conditions under which each approach is superior, problems with closed form expressions for $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^\mathcal{G}})$ will generally perform better by evaluating all $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^\mathcal{G}})$ prior to search, while when $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^\mathcal{G}})$ is solved by a machine learning training procedure, it is advantageous to use the bounding strategy in this chapter.

6.5 Preliminaries: A* Search and A* with Bounding Conflicts

We begin by reviewing A* as a method for search in graphs, and then describe A* with bounding conflicts (A*BC) as an extension. Our method further builds on existing A*BC development by providing a derivation of bounding conflicts suitable for structural search and introducing a new rule for when bounding conflicts should be computed.

6.5.1 A* Search

A* search is a complete and optimal algorithm for determining shortest paths in weighted graphs. A* performs search over a weighted graph consisting of a set of states $\mathcal{S}$ connected by a set of edges $\mathcal{E}$. Each edge $(s_i \rightarrow s_j) \in \mathcal{E}$ indicates that it is possible to travel from state s_i to state s_j, with cost $c(s_i \rightarrow s_j)$ incurred. We refer to the set of states reachable by edges from s_i as the successors of s_i. A* seeks to find the path from a given start state s_{start} to any goal state $s_{goal} \in \mathcal{S}_{goal}$ with the lowest total incurred cost. The path is constructed by sequentially traveling between states along edges in $\mathcal{E}$ and summing the costs of all edges taken.

The algorithm operates by placing states on a priority queue in order of minimization of $f(s) \triangleq g(s) + h(s)$. $g(s)$ is the cumulative cost incurred to reach state s, and $h(s)$ is an admissible heuristic function. The heuristic returns an estimate for the additional cost to be incurred by traveling from s to a goal state. The requirement that the heuristic is admissible states that $h(s) \leq \min c(s \rightarrow s_{goal})$ for all s_{goal}, where $c(s \rightarrow s_{goal})$ is the cost required to travel from s to s_{goal} and the minimum is taken over all possible paths.

Search begins with only s_{start} on the queue with $g(s_{start}) = 0$. At each iteration, the head of the queue (with lowest $g(s) + h(s)$) is popped, and the state is *expanded*, meaning its successors are generated and added to the queue. For each successor state s_{succ}, $g(s_{succ})$ is evaluated as $g(s_{succ}) = g(s) + c(s \rightarrow s_{succ})$. The queue is then re-sorted, and the process of popping and expansion repeats in a loop. Once any goal state is expanded, the path taken to that state must be the lowest cost path from s_{start} to any state in $\mathcal{S}_{goal}$

A* Search with an Expanded List

There may be multiple paths to a state s from the start state. This can result in s being placed on the queue multiple times, with different values of $g(s)$ from the different total costs to a state.

It is inefficient to repeatedly expand the same state with different costs, so it is

common in A* to maintain a list of expanded states called the *expanded list*. A state is added to the expanded list when it is expanded for the first time, and then it is never expanded again. If the state then reappears at the head of the queue, it is instead removed without successors being added to the queue.

In order for A* to remain optimal with an expanded list, it must be ensured that the first time s is expanded, it has been reached by the shortest path from s_{start} to s. If this is not true, a path from s_{start} to s_{goal} that passes through s can be made to have lower cost by replacing the part of the path from s_{start} to s with the lowest cost path to s. With the shortest path being used whenever a state is expanded, there is no better path through s, and any time s reappears on the queue it can safely be discarded without sacrificing optimality.

A sufficient condition for the lowest cost path for any state to be expanded first is to use a consistent heuristic. A consistent heuristic satisfies the additional properties that for all $s \in \mathcal{S}$ and for all successors s_{succ} of s,

$$h(s) \leq c(s \to s_{succ}) + h(s_{succ}), \tag{6.5}$$

and for any goal state, $h(s_{goal}) = 0$. If the heuristic chosen is consistent, an expanded list may be used in A* with no loss of optimality.

Maximum Reward Paths with A*

A* may instead be used to find the path with maximal reward from s_{start} to any s_{goal} if moving between states gives reward instead of cost. The algorithm runs similarly, with the priority queue constructed to maximize $g(s) + h(s)$, with $g(s)$ tracking the reward gained on the path to s. An admissible $h(s)$ is constructed to overestimate the reward of the largest reward path from s to any s_{goal}. A consistent heuristic satisfies $h(s) \geq r(s \to s_{succ}) + h(s_{succ})$, with reward function r, in addition to $h(s_{goal}) = 0$.

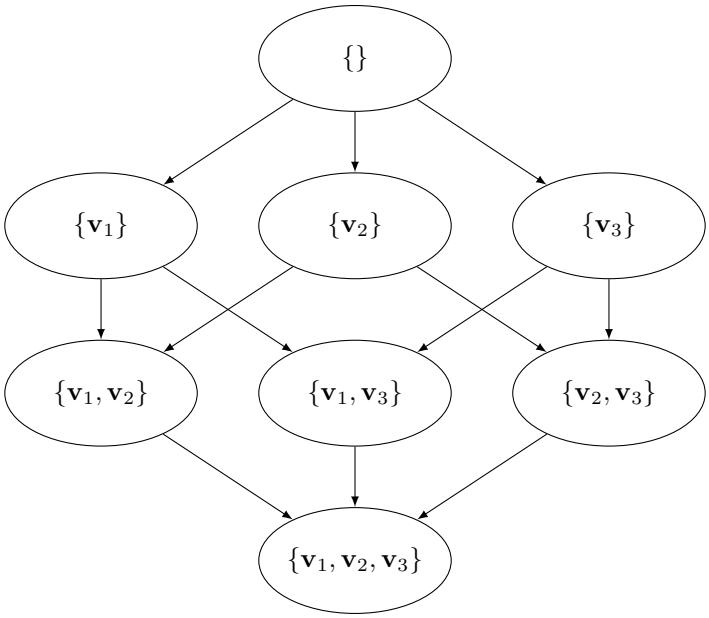


Figure 6-2: Factor graph for DAG with three variables. Duplicated from Yuan and Malone [156].

6.5.2 A* for DAG Structural Search

Our approach will use the method of converting DAG structure learning problem to a state space search problem that was developed by Yuan and Malone [156]. Their method was to search for maximum score directed acyclic graph structures using A*, where score is a penalized maximum likelihood, like in Problem 4. The method is competitive with state of the art structure learning algorithms in terms of speed, but assumes that likelihoods for all parent sets in a DAG can be evaluated. We will use the same state space as defined in this work, but we will show how to use A*BC to find the maximum score DAG without evaluating all likelihoods.

To find the maximum score DAG constructed from the set of variables $\mathcal{V} = \{\mathbf{v}_m\}$, A* search is performed over a *factor graph*. Each state in the factor graph is a subset of $\mathcal{V}$, and a path through the factor graph defines an order in which variables are added to the states along the path. An example factor graph for a DAG over 3 variables is shown in Figure 6-2. Since a path defines an ordering, it is used to represent any DAG with a topological ordering given by the variable ordering of the path.

The reward for each edge is the highest score of the newly added variable that

can be achieved by selecting a parent set from the state that the edge left. For example, when moving from state $\{\mathbf{v}_2\}$ to state $\{\mathbf{v}_1, \mathbf{v}_2\}$, a reward is given equal to the maximum of $Score(\mathbf{v}_1, \emptyset)$, and $Score(\mathbf{v}_1, \mathbf{v}_2)$. In this way, the reward of a path is equal to the reward of the maximum score DAG that is consistent with its topological ordering, and finding the maximum score DAG is achieved by finding the maximum score path through the factor graph.

Formally, the reward received by moving from a state s to $s' = s \cup \{\mathbf{v}_m\}$ is defined to be the maximum possible score that can be attained by selecting a subset of variables in s as parents of $\mathbf{v}_m$,

$$r(s \to s \cup \{\mathbf{v}_m\}) = \max_{\mathbf{v}_\Phi \subseteq s} Score(\mathbf{v}_m, \mathbf{v}_\Phi). \tag{6.6}$$

There are many DAGs consistent with a single topological ordering, and potentially multiple topological orderings for any given DAG, but the path only represents the maximum score DAG for a given topological ordering.

Search is performed from the start state $s_{start} = \{\}$, and the only goal state is $s_{goal} = \mathcal{V}$. An edge exists from s to s_{succ} if s_{succ} can be constructed by adding a single variable in $\mathcal{V}$ to s. Since the parents of each variable can only be selected from the set of variables in the preceding state, cycles cannot be introduced into the DAG by construction.

States are ordered on a queue in terms of maximization of $g(s) + h(s)$. Once the goal state is expanded, the score of the path taken is the score of the optimal DAG. $g(s)$ is computed as the accumulation of rewards along the path as in standard A*.

Multiple heuristics are proposed by Yuan and Malone. The simplest is h_{simple}, which overestimates the remaining score of variables not in s by ignoring acyclicity constraints. In this way, the optimal parent set can be selected for each variable individually,

$$h_{simple}(s) = \sum_{\mathbf{v}_m \notin s} \max_{\Psi \subseteq [D_v] \setminus \{m\}} Score(\mathbf{v}_m, \mathbf{v}_\Psi). \tag{6.7}$$

h_{simple} was proven by Yuan and Malone to be both admissible and consistent.

A more accurate heuristic is achieved by dividing the space of variables into N_{sub}

disjoint subsets $\mathcal{V}_i$ of no more than k elements, so that $\mathcal{V} = \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_{N_{sub}}$. The k-cycle heuristic is computed by enforcing acyclicity only between elements of each $\mathcal{V}_i$, and ignoring acyclicity between subsets. The static k-cycle heuristic uses subsets selected prior to the start of search, and is known to be both admissible and consistent. The dynamic k-cycle heuristic greedily selects subsets that result in the tightest bound on score each time the heuristic is calculated. The dynamic k-cycle heuristic is known to be admissible but inconsistent.

The main disadvantage of this approach for our purpose is that it requires calculation of scores of all possible parent sets prior to the start of search. In typical structure learning tasks, it is assumed that calculation of scores is fast, and the primary source of complexity comes from the large space of DAGs that needs to be evaluated. In our case, calculation of all likelihoods in the scores is the most computationally expensive part of search, making this approach infeasible. Pruning rules exist that prove that certain parent sets cannot be optimal without evaluation of their score [37, 156], but these are limited to independent discrete variables and are not applicable to our case. Use of A*BC will allow us to find the optimal DAG structure without the need to evaluate likelihoods of all possible structures.

6.5.3 A* with Bounding Conflicts

A*BC [136] is designed to improve the heuristic used in search by using information computed during candidate generation. This is done by generalizing the cost computed during testing. The approach is to represent these bounds as bounding conflicts, and to use the bounding conflicts to reorder elements on the search queue.

A*BC was developed to solve problems in which variables of the set $\mathcal{X} = \{x_1, \ldots, x_t\}$ must each be assigned elements from their respective finite domains $\mathcal{D} = \{d_1, \ldots, d_m\}$. Making the assignments $x_i = e_i$ for $e_i \in d_i$ incurs cost, and the objective is to find the assignments to all variables with the lowest total cost. The total cost does not need to be the sum of costs of individual assignments.

Search is performed over set of states where each state represents a partial assignment to the variables. A partial assignment is a set of assignments $s = \{(x_i = e_i)\}$,

which does not necessarily include an assignment to all variables in $\mathcal{X}$. Search begins at $s_{start} = \{\}$, and $\mathcal{S}_{goal}$ consists of the set of states where all variables in $\mathcal{X}$ are assigned. Edges are constructed from states with partial assignments to states where a larger set of variables has been assigned.

A* with bounding conflicts extends A* search by providing a means to learn tighter heuristics over the course of search through the use of bounding conflicts. A bounding conflict is a tuple $\gamma = (z, b)$, where z is a partial assignment of variables and $b : \{z' \,|\, z' \supseteq z\} \rightarrow \mathbb{R}$ is a function that acts on the set of larger partial assignments than z and returns a cost. The cost returned by $b(z')$ is a lower bound on the cost of any complete assignment that is a superset of z'.

The set of known bounding conflicts $\Gamma = \{\gamma\}$ can be used as a means of refining estimates of total cost $f(s)$ for any state. If s is a superset of the partial assignment z of a bounding conflict γ, then it is known that the total cost of a goal reachable from s is bounded from below as $b(s)$. Since this applies for all bounding conflicts in Γ, A*BC maintains a priority queue of states ordered in terms of the tightest cost bound that can be computed from all bounding conflicts, computed as

$$f(s, \Gamma) = \max_{(z,b) \in \Gamma \,:\, s \supseteq z} b(s). \tag{6.8}$$

Search begins with only a loose bounding function applicable to all states, and as search progresses, additional bounding conflicts are derived from evaluation of the true cost of partial assignments. The method to derive new conflicts during search is application specific. As search progresses, new conflicts are derived, the set of bounding conflicts grows, and $f(s, \Gamma)$ becomes a tighter bound on cost.

To avoid repeatedly re-evaluating the entire queue, states remain sorted using only the bounding conflicts available at the time they were queued, until they reach the head. At each iteration, search removes the head of the queue and $f(s, \Gamma)$ is recomputed using the most up to date bounding conflicts. If the state would remain at the head of the queue with its new bounded cost, it is expanded, and successor states are queued. Otherwise, the state is requeued, and the next head is considered.

Search continues until k goal states are expanded, where k is the number of best solutions desired.

We make use of the idea in A*BC of deriving bounds on cost over the course of search in order to perform efficient structural search. Bounding conflicts will be derived from the likelihoods of some structures, and used to tighten bounds on the score of structures for which the likelihood is unknown. Once a structure is found with true cost that is lower than the bounds on all other structures, it is known to be optimal, without needing to evaluate the true likelihoods of the remaining candidates.

6.6 A*BC for Structural Search

We now describe our method of using A* with bounding conflicts for DAG structural search. Our approach combines the state formulation of A* for DAG structure learning with the use of bounding conflicts derived from evaluated scores.

6.6.1 Overview of Approach

We will show that the likelihood of a variable with a given parent set can be used to place an upper bound on the score of that variable with a smaller parent set. A* search is performed over the state space described in Section 6.5.2, using these upper bounds on score to order the priority queue. As each state is expanded, the algorithm will evaluate optimal likelihoods for additional parent sets, which will tighten the bounds placed on parent sets with unevaluated likelihood. Once the goal state is expanded with a score that is exact, optimal DAG structure can be extracted from the state. This typically occurs with many fewer likelihood evaluations than performed by A*.

Our major innovations over the existing description of A*BC are a class of bounding conflicts that are applicable in different scenarios than those described in [136], and a set of rules to choose likelihoods to evaluate that are expected to generate tighter bounds. The added rules will prove to be critical to making A*BC work well in this domain.

6.6.2 Computing Bounds on Score

A*BC search is guided by bounds on scores derived from optimized likelihoods. During search, we maintain a set of computed likelihoods $\mathcal{B} = \{\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Phi)\}$, which are used to compute bounds on score for unevaluated parent sets.

We make use of the bound in the following theorem to place bounds on score. These bounds on score will be used to construct bounds on g and h in A* search from the likelihoods that have been evaluated so far. Evaluating additional likelihoods will tighten the bound. Stronger bounds are frequently used to prune the search space in structure learning, though the derived bounds are typically specific to to discrete variables [37, 156] or independent Gaussian variables, and not applicable for our problem.

Theorem 7. *Let* $\mathcal{P}_{m,\Psi} = \{\rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_\Psi) \,|\, \boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Psi}\}$ *be a set of real-valued functions parameterized by* $\boldsymbol{\theta}_m$*. Assume that for some* Ψ *and* Φ*,* $\mathcal{P}_{m,\Psi} \subseteq \mathcal{P}_{m,\Phi}$*. Then*

$$\max_{\boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Psi}} \rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_\Psi) \leq \max_{\boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Phi}} \rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_\Phi).$$

Proof. The proof follows immediately from the relation $\mathcal{P}_{m,\Psi} \subseteq \mathcal{P}_{m,\Phi}$. Since every element of $\mathcal{P}_{m,\Psi}$ is in $\mathcal{P}_{m,\Phi}$, it follows that $\max\ \mathcal{P}_{m,\Psi} \leq \max\ \mathcal{P}_{m,\Phi}$. Then

$$\begin{aligned}
\max_{\boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Psi}} \rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_\Psi) &= \max\ \mathcal{P}_{m,\Psi} \\
&\leq \max\ \mathcal{P}_{m,\Phi} \\
&= \max_{\boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Phi}} \rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_\Phi).
\end{aligned}$$

□

The following corollary shows how we may use Theorem 7 to derive bounds on likelihood for the AcyGP model. The corollary states that the optimized likelihood with parent sets that are supersets of $\mathbf{v}_{\Pi_m^\mathcal{G}}$ can be used to bound optimized likelihood with parents $\mathbf{v}_{\Pi_m^\mathcal{G}}$.

Corollary 7.1. *In the AcyGP model,*

$$\max_{\boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Psi}} p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_\Psi) \leq \max_{\boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Phi}} p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_\Phi) \;\forall\, \Psi \subseteq \Phi.$$

Proof. Define

$$\rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) = \mathbb{E}_{\mathbf{v}|\mathbf{y},\boldsymbol{\theta}^{(t,s)},\mathcal{G}^{(t)}} \left[\sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\theta}_m}(\boldsymbol{f}_m(\boldsymbol{x}_i))} \left[\log p_{\boldsymbol{\theta}_m}(y_m(\boldsymbol{x}_i) \,|\, \boldsymbol{y}_{\Pi_m^{\mathcal{G}}}(\boldsymbol{x}_i), \boldsymbol{f}_m(\boldsymbol{x}_i)) \right] \right] - \sum_{j=1}^{J_m} \mathrm{KL}(q_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j}) \,||\, p_{\boldsymbol{\theta}_{m,j}}(\mathbf{u}_{m,j}))$$

so that $\rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) = p_{\boldsymbol{\theta}_m}(\mathbf{v}_m \,|\, \mathbf{v}_{\Pi_m^{\mathcal{G}}})$. The parameters $\boldsymbol{\theta}_m$ contain the kernel hyperparameters for output m, as well as distribution parameters $\boldsymbol{\beta}_m$, parent parameters $\boldsymbol{\lambda}_{m,n,j}$, and the mean and covariance of the posterior $q_{\boldsymbol{\theta}_m}(\mathbf{u}_m)$.

In order to use Theorem 7, we must show that $\mathcal{P}_{m,\Psi} \subseteq \mathcal{P}_{m,\Phi}$. We prove this by showing that for any $\boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Psi}$, it is possible to construct a vector $\boldsymbol{\vartheta}_m \in \boldsymbol{\Theta}_{m,\Phi}$ such that $\rho_{\boldsymbol{\theta}_m}(\mathbf{v}_m, \mathbf{v}_\Psi) = \rho_{\boldsymbol{\vartheta}_m}(\mathbf{v}_m, \mathbf{v}_\Phi)$.

To construct $\boldsymbol{\vartheta}_m$ using $\boldsymbol{\theta}_m$, for each $n \in \Phi \setminus \Psi$, select the parameters $\{\boldsymbol{\lambda}_{m,n,j}\}$ such that $t_{\boldsymbol{\lambda}_{m,n,j}}(\cdot) = 0$. This choice effectively zeroes out the influence of parent n on output m. Then, select the kernel hyperparameters, $\boldsymbol{\beta}_m$, $\{\boldsymbol{\lambda}_{m,n,j}\}$ with $n \in \Psi$, and the variational posterior mean and covariance identically to $\boldsymbol{\theta}_m$. This fully recovers the structural equation model of the AcyGP model with parent set $\mathbf{v}_\Psi$.

Since this process can be performed for any $\boldsymbol{\theta}_m \in \boldsymbol{\Theta}_{m,\Psi}$, we have $\mathcal{P}_{m,\Psi} \subseteq \mathcal{P}_{m,\Phi}$. The corollary then follows from direct application of Theorem 7. □

As an example, consider the sketch of the heterogeneous AcyGP model from Chapter 5, which we repeat in Figure 6-3. An AcyGP over the same variables could be defined with no edges, in which case we would have $\alpha_{S_p}(\boldsymbol{x}) = f_{S_p}(\boldsymbol{x})$. Corollary 7.1 states that we can recover any possible set of parameters in the AcyGP with no edges by selecting $\lambda_{S_p,M_p,0} = \lambda_{S_p,M_p,1} = \lambda_{S_p,B_p,0} = \lambda_{S_p,B_p,1} = 0$, which makes the edges have no effect. Other parameters in the model can be selected identically to the AcyGP with no edges.

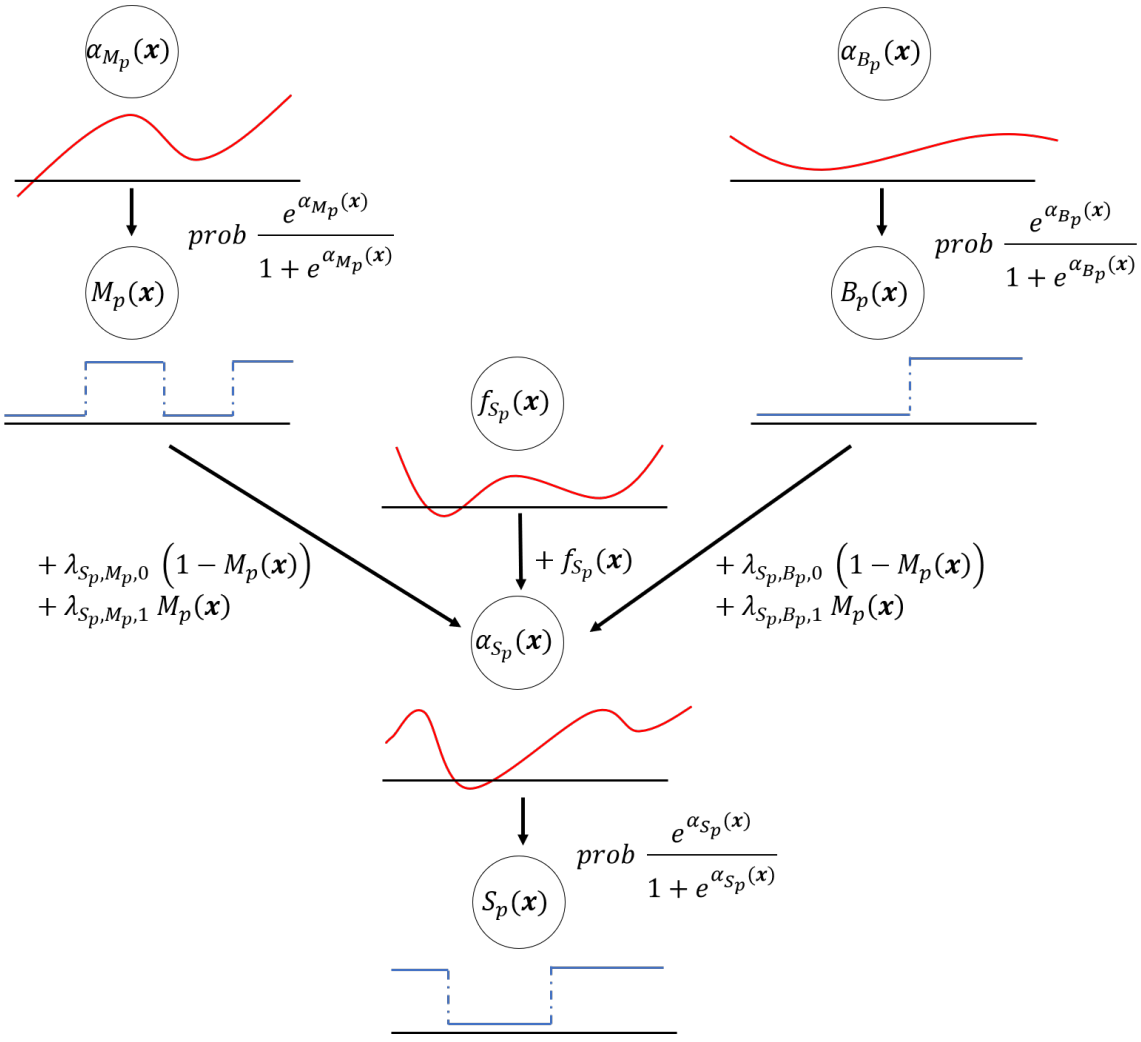


Figure 6-3: Sketch of a heterogeneous AcyGP.

Using Corollary 7.1, $Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}})$ is bounded using the optimized likelihood of any superset of $\Pi_m^{\mathcal{G}}$ and the true complexity cost $c_m(\Pi_m^{\mathcal{G}})$. During search, we can make use of any likelihood in the known set $\mathcal{B}$. We define the tightest available upper bound on score for the known set of likelihoods as $\overline{Score}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathcal{B})$, computed as

$$\overline{Score}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathcal{B}) := \min_{\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Phi) \in \mathcal{B} : \Phi \supseteq \Pi_m^{\mathcal{G}}} \hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Phi) - c_m(\Pi_m^{\mathcal{G}}) \tag{6.9}$$

Within our search algorithm, we will compute bounds on the functions $g(s)$ and $h(s)$ by using $\overline{Score}$. $\overline{Score}$ is the best bound we have available, and will be used to guide search.

6.6.3 Algorithm Description

An overview of A*BC for DAG structure learning is given in Algorithm 17. We use the same state space as in A* search, so that each state s is a subset of the variables $\mathcal{V}$. A path through the state space defines a topological ordering.

Search starts with a start state {} on the queue on line 1. States are repeatedly popped from the head of the queue, and either requeued or expanded until a goal state is reached with a tight bound.

In our A*BC algorithm, we aim to approximate the functions $g(s)$ and $h(s)$ used in A* for DAG structure learning with the previously derived upper bounds on score. This allows us to approximate the reward that can be achieved by a path through a state s, using information from the likelihoods we have evaluated. States are sorted on the priority queue in order of maximization of $\bar{g}(s, \mathcal{B}) + \bar{h}(s, \mathcal{B})$, which respectively bound $g(s)$ and $h(s)$ using the set of evaluated likelihoods $\mathcal{B}$. $\bar{g}$ and $\bar{h}$ are computed by using $\overline{Score}$ in the place of $Score$ for the definitions of $g(s)$ and $h_{simple}(s)$ in A* for DAG search described in Section 6.5.2.

To ensure that restrictions on possible parent sets expressed through the sets $\{\Xi_m\}$ are respected, we add the following additional restriction. The reward of a path is equal to the maximum score DAG that has two properties, a) it is consistent with the topological ordering defined by the path, and b) it has the parent set of $\mathbf{v}_m$ as an

element of Ξ_m for all m. This leads to the following definitions for $\bar{g}$ and $\bar{h}$,

$$\bar{g}(s, \mathcal{B}) = \sum_{\mathbf{v}_{m_r} \in T(s)} \max_{\substack{\Psi \in \Xi_{m_r} \\ \Psi \subseteq \{m_1, \ldots, m_{r-1}\}}} \overline{Score}(\mathbf{v}_{m_r}, \mathbf{v}_\Psi, \mathcal{B}) \tag{6.10}$$

$$\bar{h}(s, \mathcal{B}) = \sum_{\mathbf{v}_m \notin s} \max_{\substack{\Psi \in \Xi_m \\ \Psi \subseteq [D_y] \setminus \{m\}}} \overline{Score}(\mathbf{v}_m, \mathbf{v}_\Psi, \mathcal{B}), \tag{6.11}$$

where $T(s) = (\mathbf{v}_{m_1}, \mathbf{v}_{m_2}, \ldots, \mathbf{v}_{m_{|s|}})$ is a topological ordering over the variables in s derived from the path taken to s.

Like in Timmons and Williams [136], to avoid repeatedly re-evaluating the entire queue when a new likelihood has been evaluated, states remain sorted using only the set $\mathcal{B}$ available at the time they were queued, until they reach the head. At each iteration, search removes the head of the queue on line 3 and $\bar{g}(s, \mathcal{B}) + \bar{h}(s, \mathcal{B})$ is recomputed using the most up to date set of optimized likelihoods on line 4. If a state does not remain at the head of the queue with its new bounded reward, the state is requeued on line 6, and the next head is considered. If the state would remain at the head of the queue with its new bounded reward, it is expanded on line 14, additional likelihoods are evaluated with the procedure *EvaluateLikelihoods*, and successor states are queued. If state s' would be a successor of state s as $s' = s \cup \{\mathbf{v}_m\}$, but there is no set $\Pi_m \subseteq s$ such that $\Pi_m \in \Xi_m$, then the path would not be able to represent any DAG under this restriction. As a result, s' is not generated as a successor to s.

Search starts with all likelihoods $\{\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{[D_y] \setminus \{m\}}\}$ evaluated. These largest possible parent sets are evaluated so that a bound exists for every possible parent set in the problem. Since no larger parent sets exist to place finite bounds on $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{[D_y] \setminus \{m\}})$, the bounded score for these optimized likelihoods is not finite if they have not been evaluated. As a result, DAGs with full parent sets will have infinite bounded score, and will always appear at the head of the queue until they are evaluated in any best first search procedure. Therefore, computing them at the start of search is not wasteful.

Treatment of possible goal states are handled on line 7. A goal state is a state

to which all variables $\mathcal{V}$ have been added, so that a path to it represents a full DAG over the variables in our problem. The goal state is only expanded once $\bar{g}(s_{goal}, \mathcal{B}) = g(s_{goal})$, meaning that the optimized likelihoods of the DAG with the best bounded score represented by s_{goal} have been computed exactly, as checked on line 8. This is necessary, because if $\bar{g}(s_{goal}) \neq g(s_{goal})$, it is not necessarily true that the path to s_{goal} constructs the optimal score DAG. Another state s' on the true optimal path to s_{goal} may exist on the queue with a tighter likelihood bound, so that $\bar{g}(s', \mathcal{B}) + \bar{h}(s', \mathcal{B}) \leq \bar{g}(s_{goal})$, causing s_{goal} to be expanded first. If $\bar{g}(s, \mathcal{B})$ is not exact, additional likelihoods are computed from its maximum bounded score parent sets, and it is requeued. A*BC terminates once a goal state is expanded, returning the DAG that optimized the score of the state.

Algorithm 17: A*BC Structural Search

Input : Optimized likelihood set $\mathcal{B} = \{\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{[D_y]\setminus\{m\}})\}$
Output: Maximum score DAG $\mathcal{G}$

1 Initialize Q with an empty state
2 **loop** while goal state not expanded
3 $\quad s \leftarrow pop(Q)$
4 $\quad \bar{g} \leftarrow \bar{g}(s, \mathcal{B})$, $\bar{h} \leftarrow \bar{h}(s, \mathcal{B})$ `// Recompute with latest bounding conflicts`
5 $\quad$ **if** $\bar{g} + \bar{h}$ is no longer highest reward in Q **then**
6 $\quad\quad$ Requeue s with priority $\bar{g} + \bar{h}$, continue
7 $\quad$ **else if** s is a goal state with highest bounded score DAG $\mathcal{G}$ **then**
8 $\quad\quad$ **if** $\exists$ $\mathbf{v}_m$ such that $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi^{\mathcal{G}}_m}) \notin \mathcal{B}$ **then** `// Bound not tight`
9 $\quad\quad\quad$ $\mathcal{B} \leftarrow \mathcal{B} \cup \{\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi^{\mathcal{G}}_m})\}$
10 $\quad\quad\quad$ Queue s with priority $\bar{g}(s, \mathcal{B}) + \bar{h}(s, \mathcal{B})$, continue
11 $\quad\quad$ **else return** $\mathcal{G}$
12 $\quad$ **else for** $\mathbf{v}_m \notin s$ **do** `// Make successor states`
13 $\quad\quad$ $\mathcal{B} \leftarrow EvaluateLikelihoods(s, \mathbf{v}_m, \mathcal{B})$
14 $\quad\quad$ Queue $s \cup \{\mathbf{v}_m\}$ with priority $\bar{g}(s \cup \{\mathbf{v}_m\}, \mathcal{B}) + \bar{h}(s \cup \{\mathbf{v}_m\}, \mathcal{B})$, continue
15 $\quad$ **end**

6.6.4 Rules for Computation of Optimized Likelihood

A major difference between our approach and the existing A*BC implementation is that it is easy to generate bounds by computing likelihoods at any time. Whereas

bounds naturally arise [136] when evaluating the cost for complete assignments, in our case, any optimized likelihood could be evaluated at any point in search in order to produce a new bound. In Algorithm 17, the set of optimized likelihoods $\mathcal{B}$ are evaluated as part of the routine *EvaluateLikelihoods*, which we now describe.

EvaluateLikelihoods produces bounds for a set of parent assignments that is at the head of the queue when a state is expanded. In this way, parent sets that are likely to be optimal, as determined by progress through the A* search space, have their likelihoods evaluated, producing tighter bounds and making their optimality less likely. The parent sets that are evaluated are either the best known parent set, or another that is expected to remove the need to evaluate the current best known parent set.

In order to decide which likelihoods to evaluate when, consider that unlike the application of A*BC developed by Timmons and Williams [136], where a bounding conflict is used to bound any extension of an assignment to variables, our bound uses a given assignment to the parent set of $\mathbf{v}_m$ to bound the reward of other parent set assignments of the same variables $\mathbf{v}_m$. Further, the bound is only applicable to parent sets that are subsets of evaluated likelihoods. This raises a unique tradeoff to be addressed during search. On one hand, it is advantageous to evaluate $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}})$ for relatively large parents sets $\mathbf{v}_{\Pi_m^{\mathcal{G}}}$ in order to place bounds on the scores of a large number of DAGs. On the other hand, optimized likelihoods for smaller parent sets give tighter bounds for their subsets. Search must evaluate likelihoods that are large enough to bound multiple possible DAGs, while ensuring the bounds are tight enough to push suboptimal DAGs further down the search queue.

An algorithmic description for *EvaluateLikelihoods* is given in Algorithm 18. When a state s is expanded, each successor is generated by adding $\mathbf{v}_m \notin s$ to the set of variables currently contained in s. $\bar{g}(s \cup \{\mathbf{v}_m\}, \mathcal{B})$ is computed for the successor

state as

$$\bar{g}(s \cup \{\mathbf{v}_m\}, \mathcal{B}) = \bar{g}(s, \mathcal{B}) + \overline{Score}(\mathbf{v}_m, \mathbf{v}_\Psi, \mathcal{B}) \tag{6.12}$$

$$\mathbf{v}_\Psi = \underset{\mathbf{v}_\Phi \subseteq s,\ \Phi \in \Xi_m}{\arg\max}\ \overline{Score}(\mathbf{v}_m, \mathbf{v}_\Phi, \mathcal{B}). \tag{6.13}$$

Unless $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$ has been evaluated, $\overline{Score}(\mathbf{v}_m, \mathbf{v}_\Psi, \mathcal{B})$ is inexact. We can tighten $\bar{g}(s \cup \{\mathbf{v}_m\}, \mathcal{B})$ by evaluating $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$, or alternatively, evaluating $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)$ for some non-empty $\mathbf{v}_\Omega$. The latter option is preferred, because it is possible that evaluation of $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)$ tightens $\overline{Score}(\mathbf{v}_m, \mathbf{v}_\Psi, \mathcal{B})$ enough that the new state $s \cup \{\mathbf{v}_m\}$ never reaches the head of the queue. If this occurs, then $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$ never needs to be evaluated, saving the evaluation time. Evaluating $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)$ has the additional utility that it may be used to derive bounds on other subsets of $\mathbf{v}_\Psi \cup \mathbf{v}_\Omega$, while $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$ may only be used to derive bounds on subsets of $\mathbf{v}_\Psi$. The choice of $\mathbf{v}_\Omega$ is intended to lower $\overline{Score}(\mathbf{v}_m, \mathbf{v}_\Psi, \mathcal{B})$ as much as possible, while still generating a bound applicable for other parent sets. This is done in order to move the newly generated successor state deeper into the search queue, so that the optimum DAG is found without considering the state again and needing to evaluate more likelihoods.

Our strategy for selecting $\mathbf{v}_\Omega$ is based on the heuristic observation that in sparse DAGs, likelihood with any single parent in the optimal parent set is often (but not always) significantly increased over likelihood when using a single parent not in the optimal set. That is,

$$\hat{\rho}(\mathbf{v}_m, \{\mathbf{v}_n\}) > \hat{\rho}(\mathbf{v}_m, \{\mathbf{v}_l\}), \quad n \in \Pi_m^{\mathcal{G}^*}, l \notin \Pi_m^{\mathcal{G}^*}.$$

To choose $\mathbf{v}_\Omega$, we combine k variables in s that result in *lowest* likelihood when used as the only parent of $\mathbf{v}_m$, and are not already in $\mathbf{v}_\Psi$. In this way, elements of $\mathbf{v}_\Omega$ are unlikely to be part of the optimal parent set, and we anticipate that $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)$ is not significantly larger than $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$. It frequently occurs that after $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)$ is evaluated, $\mathbf{v}_\Psi$ is no longer the best candidate parent set for output $\mathbf{v}_m$. In this case, no new optimized likelihoods are computed, and search proceeds using the new

best candidate parent set. In the case where $\mathbf{v}_\Psi$ remains the best candidate parent set, for the new state, the likelihood $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$ is evaluated.

Algorithm 18: EvaluateLikelihoods

Input : Expanded state s, variable to add $\mathbf{v}_m$, optimized likelihood set $\mathcal{B}$
Output: Larger optimized likelihood set $\mathcal{B}'$ with additional evaluated parent sets of $\mathbf{v}_m$

$\mathbf{v}_\Psi \leftarrow \arg\max_{\mathbf{v}_\Phi \subseteq s} \overline{Score}(\mathbf{v}_m, \mathbf{v}_\Phi, \mathcal{B})$ `// Best parent set under bounds`
$\mathbf{v}_\Omega \leftarrow k$ values of $\mathbf{v}_n \in s \setminus \mathbf{v}_\Psi$ with lowest $\hat{\rho}(\mathbf{v}_m, \{\mathbf{v}_n\})$ in $\mathcal{B}$
$\mathcal{B}' \leftarrow \mathcal{B} \cup \{\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)\}$
if $\arg\max_{\mathbf{v}_\Phi \subseteq s} \overline{Score}(\mathbf{v}_m, \mathbf{v}_\Phi, \mathcal{B}) = \mathbf{v}_\Psi$ **then**
  $\mathcal{B}' \leftarrow \mathcal{B}' \cup \{\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)\}$
end
return $\mathcal{B}$'

While *EvaluateLikelihoods* could evaluate up to two likelihoods instead of only evaluating $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$, it actually causes significantly fewer optimized likelihoods to be computed over the course of search. If the algorithm only evaluated $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$, that likelihood cannot be used to bound $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)$. When the bound for $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$ is loose, meaning it is computed using the optimized likelihood of a much larger parent set, it is frequently the case that $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)$ is loosely bounded by a similarly large parent set. This then makes $\mathbf{v}_\Psi \cup \mathbf{v}_\Omega$ a new candidate optimal parent set, which will have to be evaluated during the course of search regardless. In contrast, by choosing to evaluate $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)$ first, we can use the result to refine the bound on $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$. If $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)$ is low, it may lower the bound on $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$ so that $\mathbf{v}_\Psi$ never becomes a candidate optimal parent set for $\mathbf{v}_m$, so that the true value of $\hat{\rho}(\mathbf{v}_m \mid \mathbf{v}_\Psi)$ is never evaluated. This explains why it is important to evaluate larger parent sets before smaller parent sets. Our experimental results confirm this intuition, and we see significantly fewer likelihoods evaluated under our proposed strategy, compared to evaluating the likelihood of the best parent set.

When k is too low, $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi \cup \mathbf{v}_\Omega)$ is a bound for fewer candidate sets. When k is too high, the bound produced is too loose, and $\mathbf{v}_\Psi$ remains the best parent set. Generally, optimized likelihood increases more significantly with additional variables when Ψ is smaller, so we increase k as a function of $|\Psi|$. Empirically, we find strong

performance using $k = 1$ when $|\Psi| \leq 2$ and $k = 2$ otherwise.

This procedure is applied in the following example.

Example 3. *Consider an optimal DAG search problem where for all m, $c_m(\Psi) = |\Psi|$. Assume that search expands the state $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ and generates the successor state $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$. At the time of expansion, $\mathcal{B}$ contains the following optimized likelihoods:*

$$\hat{\rho}(\mathbf{v}_4, \mathbf{v}_\emptyset) = -55$$
$$\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1\}}) = -50$$
$$\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{2\}}) = -11.5$$
$$\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,2,3\}}) = -10.$$

The maximum bounded score parent set for $\mathbf{v}_4$ is $\{\mathbf{v}_3\}$, with bounded score

$$\overline{Score}(\mathbf{v}_4, \mathbf{v}_{\{3\}}, \mathcal{B}) = \hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,2,3\}}) - c_4(\{3\}) = -11.$$

We observe that $\mathbf{v}_4$ is almost fully described by dependence on $\mathbf{v}_{\{2\}}$, because $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{2\}})$ is almost as large as $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,2,3\}})$. Since the optimized likelihood of $\mathbf{v}_4$ changes little with the inclusion of $\mathbf{v}_3$, it is reasonable to expect that $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{3\}})$ is low, and that the bound above is loose.

If this expectation is true, and $\mathbf{v}_4$ is almost fully described by dependence on $\mathbf{v}_{\{2\}}$, then we also expect that $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,3\}})$ is low. Furthermore, $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,3\}})$ must be larger than $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{3\}})$, so calculating $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,3\}})$ to be low proves that $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{3\}})$ must also be low. In this case, our approach identifies $\mathbf{v}_1$ as the variable with the lowest likelihood when used as the only parent on $\mathbf{v}_4$, and first evaluates $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,3\}})$.

If we compute $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,3\}})$ and find that $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,3\}}) = -40$, we can compute the following bound,

$$\overline{Score}(\mathbf{v}_4, \mathbf{v}_{\{3\}}, \mathcal{B}) = \hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,3\}}) - c_4(\{3\}) = -41.$$

This bound is below the known score $Score(\mathbf{v}_4, \mathbf{v}_{\{2\}}) = -12.5$*, we know that* $\mathbf{v}_{\{3\}}$ *cannot be the optimal parent set for* $\mathbf{v}_4$ *when* $\mathbf{v}_{\{2\}}$ *is available. The likelihood* $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{3\}})$ *then never needs to be computed explicitly.*

To see that this is not wasteful, assume instead that we compute $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{3\}})$ *and find* $\hat{\rho}(\mathbf{v}_4 \,|\, \mathbf{v}_{\{3\}}) = -43$*. Then the maximum bounded score parent set for* $\mathbf{v}_4$ *is* $\mathbf{v}_{\{1,3\}}$*, with bounded score*

$$\overline{Score}(\mathbf{v}_4, \mathbf{v}_{\{1,3\}}, \mathcal{B}) = \hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,2,3\}}) - c_4(\{1,3\}) = -12.$$

Under this loose bound, $\mathbf{v}_{\{1,3\}}$ *is the parent set with the highest bounded score, and it is highly likely that* $\hat{\rho}(\mathbf{v}_4, \mathbf{v}_{\{1,3\}})$ *still needs to be computed later in search.*

6.7 Proof of Optimality

We now present a complete proof that the DAG returned by A*BC has the highest score of all possible DAGs. The proof is performed by contradiction. We show that if a DAG with higher score exists when the goal state is expanded, then a state with higher priority exists on the queue. But this higher priority state must be expanded before the goal state, so cannot be on the queue, yielding a contradiction. We first begin with two lemmas. Lemma 2 establishes that the score of an expanded goal state is exact.

Central to the proof is the fact that there are no loops in the graph that is being searched, otherwise a path to a state can accumulate reward for the parents of a single index multiple times. The fact that there are no loops is satisfied by construction in our problem.

Lemma 2. *Upon expansion, the reward of goal state* s *is equal to the exact score of the maximum score DAG with topological ordering* s*.*

Proof. Denote the known bounding conflicts at the time of expansion of s as $\mathcal{B}$, and denote the DAG with highest known *bounded* score at the time of expansion with topological ordering s as $\mathcal{G}$. By definition, $\mathcal{G}$ is used to score s at expansion. To prove

the lemma, we prove that the score of s at expansion is the exact score of $\mathcal{G}$, and that there is no DAG $\mathcal{G}'$ with topological ordering s with higher score.

By the description of our algorithm, s is only expanded once all likelihoods have been computed for $\mathcal{G}$. This implies that for all $\mathbf{v}_m$, $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) \in \mathcal{B}$ when s is expanded. All outputs $\mathbf{v}_m$ are elements of s, so the heuristic is zero, and the bounded score of s at expansion is given by

$$\begin{aligned}
\bar{g}(s, \mathcal{B}) &= \sum_m \overline{Score}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathcal{B}) \\
&= \sum_m \min_{\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Phi) \in \mathcal{B} \,:\, \Phi \supseteq \Pi_m^{\mathcal{G}}} \hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Phi) - c_m(\Pi_m^{\mathcal{G}}) \\
&= \sum_m \hat{\rho}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) - c_m(\Pi_m^{\mathcal{G}}) \\
&= \sum_m Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}).
\end{aligned}$$

The third equality follows from Theorem 7.1 and the fact that the exact likelihoods are in the set of bounding conflicts. This shows that the reward of s is the exact score of $\mathcal{G}$.

Now assume for contradiction that there exists another DAG $\mathcal{G}'$ with topological ordering s with higher true score than $\mathcal{G}$. Then

$$\begin{aligned}
\sum_m \overline{Score}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathcal{B}) &= \sum_m Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) \\
&\leq \sum_m Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}'}}) \\
&\leq \sum_m \overline{Score}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}'}}, \mathcal{B}).
\end{aligned}$$

Therefore $\mathcal{G}$ cannot be the DAG with highest known bounded score with topological ordering s, resulting in the contradiction. □

Lemma 3 states that the priority $\bar{g}(s, \mathcal{B}) + \bar{h}(s, \mathcal{B})$ of a state bounds the true score of any DAG with topological ordering consistent with the path taken to s.

Lemma 3. *Consider any search state* $s = \{\mathbf{v}_{m_1}, \ldots, \mathbf{v}_{m_r}\}$ *reached by a path that*

added variables in the order $T(s) = (\mathbf{v}_{m_1}, \ldots, \mathbf{v}_{m_r})$. *For any DAG* $\mathcal{G}$ *over all* $\mathbf{v}_m$ *with topological ordering that begins with* $T(s)$*, and any set of bounding conflicts* $\mathcal{B} \supseteq \{\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{[D_y]\setminus\{m\}})\}$*, then*

$$\sum_{\mathbf{v}_m} Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) \leq \bar{g}(s, \mathcal{B}) + \bar{h}(s, \mathcal{B}).$$

Proof. The score of the DAG may be decomposed as

$$\sum_{\mathbf{v}_m \in s} Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) + \sum_{\mathbf{v}_m \notin s} Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}).$$

The two elements of the expression may be bounded by $\bar{g}(s, \mathcal{B})$ and $\bar{h}(s, \mathcal{B})$ respectively. Since $\mathcal{G}$ has topological ordering that begins with $T(s)$, it must be that for all $\mathbf{v}_{m_r} \in s, \Pi_m^{\mathcal{G}} \subseteq \{m_1, \ldots, m_{r-1}\}$. Then, since $\mathcal{B} \supseteq \{\hat{\rho}(\mathbf{v}_m, \mathbf{v}_{[D_y]\setminus\{m\}})\}$, $\mathcal{B}$ must contain a likelihood of $\mathbf{v}_m$ evaluated with a superset of $\Pi_m^{\mathcal{G}}$ for all m, and an upper bound on score must exist for all outputs and parent sets. It follows that

$$\begin{aligned}
\sum_{\mathbf{v}_m \in s} Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) &\leq \sum_{\mathbf{v}_m \in s} \overline{Score}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathcal{B}) \\
&\leq \sum_{\mathbf{v}_m \in s} \max_{\Psi \subseteq \{m_1, \ldots, m_{r-1}\}} \overline{Score}(\mathbf{v}_m, \mathbf{v}_\Psi, \mathcal{B}) \\
&= \bar{g}(s, \mathcal{B}).
\end{aligned}$$

Similarly,

$$\begin{aligned}
\sum_{\mathbf{v}_m \notin s} Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) &\leq \sum_{\mathbf{v}_m \notin s} \overline{Score}(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}, \mathcal{B}) \\
&\leq \sum_{\mathbf{v}_m \in \mathcal{P}} \max_{\Psi \subseteq [D_y]\setminus\{m\}} \overline{Score}(\mathbf{v}_m, \mathbf{v}_\Psi, \mathcal{B}) \\
&= \bar{h}(s, \mathcal{B}),
\end{aligned}$$

Summing the two expressions completes the proof. □

We now prove the optimality of our A*BC search procedure, as stated by the following theorem.

Theorem 8. *Once a goal state is expanded in Algorithm 17, its score is the score of the optimal DAG.*

Proof. Denote the expanded goal state by s, the topological ordering implied by the path as $T(s)$, and the known set of likelihoods at the time of expansion as $\mathcal{B}$. In addition, let the DAG with highest bounded score with topological ordering $T(s)$ be denoted by $\mathcal{G}$.

From lemma 2 we know that s has priority at expansion equal to the exact (not bounded) score of $\mathcal{G}$, and that no other DAG with topological ordering given by $T(s)$ has higher score. Next, we prove that no other DAG can have a higher score than $\mathcal{G}$, regardless of ordering.

Assume for contradiction that another DAG $\mathcal{G}'$ exists with higher score than $\mathcal{G}$. $\mathcal{G}'$ must have a different topological ordering $T(s') \neq T(s)$. Whenever a state is expanded in A*BC, successor states are added to the queue with all possible outputs added to the expanded state. As a result, there must be another state s'' on the queue consisting of first r elements of $T(s')$.

Since s'' is not at the head of the queue, its reward was computed with a set of known likelihoods $\mathcal{B}'' \subseteq \mathcal{B}$. Then

$$\begin{aligned}
\bar{g}(s, \mathcal{B}) &= \sum_{\mathbf{v}_m} Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}}}) \\
&\leq \sum_{\mathbf{v}_m} Score(\mathbf{v}_m, \mathbf{v}_{\Pi_m^{\mathcal{G}'}}) \\
&\leq \bar{g}(s'', \mathcal{B}'') + \bar{h}(s'', \mathcal{B}''),
\end{aligned}$$

where the final inequality follows from lemma 3. Use of the lemma is valid, because bounding conflicts for the largest parent sets of each output are computed at the start of A*BC. However, the result above implies that s cannot be at the head of the queue and expanded, because s'' has larger bounded score, resulting in a contradiction. □

6.8 Using an Expanded List

As discussed in Section 6.5.1, in A* the use of an expanded list greatly reduces search complexity by ensuring that the same state is not repeatedly placed on the queue a large number of times. In order for search to remain optimal, the first time any state s is expanded, it must have been reached by the highest reward path to s. In A*, use of a consistent heuristic is a sufficient condition for this property.

This property no longer holds when search is performed with priority $\bar{g}(s, \mathcal{B}) + \bar{h}(s, \mathcal{B})$, even when $\bar{h}$ is consistent. As a counterexample, consider some s on the queue, reached by suboptimal path $T(s)$ while state s' reached by part of the optimal path to s is also on the queue. Consistency of $\bar{h}$ tells us that

$$g(s) + \bar{h}(s, \mathcal{B}) \leq g(s') + \bar{h}(s', \mathcal{B}).$$

This is not sufficient to prove that s' is expanded before s in our algorithm, because priorities use $\bar{g}$ instead of g.

Even if the priorities are evaluated with the same set of likelihoods, so that s and s' are queued with respective priorities $\bar{g}(s, \mathcal{B}) + \bar{h}(s, \mathcal{B})$ and $\bar{g}(s', \mathcal{B}) + \bar{h}(s', \mathcal{B})$, it is feasible that $\mathcal{B}$ results in a loose bound for $\bar{g}(s, \mathcal{B})$ and a tight bound for $\bar{g}(s', \mathcal{B})$. This can result in

$$g(s) + \bar{h}(s, \mathcal{B}) \leq g(s') + \bar{h}(s', \mathcal{B}) \leq \bar{g}(s', \mathcal{B}) + \bar{h}(s', \mathcal{B}) \leq \bar{g}(s, \mathcal{B}) + \bar{h}(s, \mathcal{B}).$$

This results in s being expanded before s', even though it has been reached by a suboptimal path.

However, it is possible to define another condition which is sufficient to determine that the maximum reward path to any state has been found, and allows us to construct an expanded list for our search algorithm. This condition is described in the following theorem. Intuitively, this theorem states that a state can be added to the expanded list if it is expanded at a time when the reward up to that state is known exactly, so that any bounds on it are tight.

Theorem 9. *Assume that $\bar{h}$ is a consistent heuristic for fixed $\mathcal{B}$, meaning that for any state s and any path from state s' to s, $\bar{h}(s', \mathcal{B}) \geq r(s' \to s) + \bar{h}(s, \mathcal{B})$. Further assume that $\bar{h}$ can only be reduced with larger $\mathcal{B}$, so that for $\mathcal{B}' \subseteq \mathcal{B}$, $h(s, \mathcal{B}') \geq h(s, \mathcal{B})$. If state s is ever expanded with $\bar{g}(s, \mathcal{B}) = g(s)$, then s has been expanded on the maximum possible reward path.*

Proof. Denote the reward obtained on the highest score path to state s as $g^*(s)$, and the bound for this path as $\bar{g}^*(s, \mathcal{B})$. Assume for contradiction that s has been expanded with set of optimized likelihoods $\mathcal{B}$ on a path with reward $g(s) < g^*(s)$, but that s has never been expanded on the highest reward path. Then there exists a state s' on the queue with priority evaluated with $\mathcal{B}' \subseteq \mathcal{B}$ that is reached the maximum reward path to s' that is a subset of the optimal path to s. Then

$$\begin{aligned}
\bar{g}^*(s', \mathcal{B}') + \bar{h}(s', \mathcal{B}') &\geq g^*(s') + \bar{h}(s', \mathcal{B}') \\
&\geq g^*(s') + r(s' \to s) + \bar{h}(s, \mathcal{B}') \\
&= g^*(s) + \bar{h}(s, \mathcal{B}') \\
&\geq g^*(s) + \bar{h}(s, \mathcal{B}) \\
&> g(s) + \bar{h}(s, \mathcal{B}).
\end{aligned}$$

The first line follows from $\bar{g}^*$ being an overestimate for g^*, the second line results from the consistency of $\bar{h}$, the fourth line from $\bar{h}$ being tightened by larger $\mathcal{B}$, and the final line results from the optimality of $g^*(s)$.

Since s was queued with priority $g(s) + \bar{h}(s, \mathcal{B})$, the inequality above shows that s' must have had a higher priority than s. Then s cannot be have been at the head of the queue and expanded, resulting in a contradiction. □

Theorem 9 implies that once a state s has been expanded with a reward to the state that is exact, then any future paths to s are suboptimal and can be ignored in search. During search, we maintain an expanded list, and add states to the list once they are expanded with $\bar{g}(s, \mathcal{B}) = g(s)$. Any time a state on the expanded list would be expanded, it is instead ignored. To check that $\bar{g}(s, \mathcal{B}) = g(s)$ is satisfied, we verify

that for all $\mathbf{v}_{m_r} \in T(s)$,

$$\hat{\rho}(\mathbf{v}_{m_r}, \mathbf{v}_\Phi) \in \mathcal{B} \text{ where } \Phi = \underset{\Psi \subseteq \{m_1,\ldots,m_{r-1}\}}{\arg\max} \overline{Score}(\mathbf{v}_{m_r}, \mathbf{v}_\Psi, \mathcal{B}). \tag{6.14}$$

Note that Theorem 9 does *not* say that when s is expanded, it has been reached by the maximum reward path. It is possible that s has been previously reached on the maximum score path and expanded with inexact $\bar{g}$. Theorem 9 only states that any future times that s is reached, the state may be ignored.

6.9 Comparison of Efficiency Between A* and A*BC

The A*BC procedure presented in this chapter is designed for scores that are expensive to evaluate. Compared to A* DAG search, our algorithm evaluates fewer score functions, resulting in significant time savings. Our approach is not recommended for use when scores are computationally cheap to evaluate, and it will typically lead to slower search in this case.

Our A*BC procedure orders the priority queue based on bounds on both reward received and the heuristic $\bar{g}(s, \mathcal{B}) + \bar{h}(s, \mathcal{B})$, which may be quite loose when few optimized likelihoods have been evaluated. Compared to A* search that orders the priority queue by $g(s) + h(s)$, the looser bounds in A*BC means that more states will appear at the head of the queue as search progresses, and more states will expanded in total. By delaying computation of scores, A*BC acts with limited information that is available at the start of search of A*. As a result, a less efficient search over states is effectively used by A*BC and more states are expanded than in A*, but the advantage is that fewer likelihoods will be evaluated.

In problems where likelihoods are computationally expensive to compute, it is worthwhile to accept more complex search expanded with fewer likelihoods evaluated. However, when exact scores are easily computed, there is little to be gained by delaying computation of optimized likelihoods, and the additional complexity of search using A*BC is disadvantageous.

6.10 Markov Equivalence Classes in AcyGP Structural Search

In this section, we will explain why we do not perform search over the smaller space of Markov equivalence classes for DAGs, as is performed by some structure learning algorithms. Instead, we search over the space of DAGs.

Multiple directed acyclic graphs can encode the same set of conditional independence statements [142], and therefore are capable of representing the same set of distributions. Such graphs are said to be *Markov equivalent*, and a set of DAGs that are all Markov equivalent is known as a *Markov equivalence class*. Since Markov equivalent DAGs represent the same set of distributions, many score functions including BIC give the same score to all DAGs in the same Markov equivalence class [29], which is known as *score equivalence*.

Score equivalence is the foundation of algorithms such as Greedy Equivalent Search [30], which searches over the smaller space of Markov equivalence classes, rather than the full space of DAGs. Score equivalence can be shown to hold for DAGs over specific parameterized distributions, such as discrete or Gaussian DAGs, allowing efficient search over the space of equivalence classes to be used.

Our approach does not perform search over the space of Markov equivalence classes. This is because our primary application is the AcyGP model, and AcyGPs are parameterized such that two AcyGPs with identical conditional independence relationships between outputs do not represent the same distributions. AcyGPs are therefore not score equivalent under scores such as BIC.

As an example, there are two possible connected DAG structures between two variables, $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2\}$, as shown in Figure 6-4. The DAGs in Figure 6-4 are Markov equivalent; Figures 6-4a and 6-4b represent the same distributions. Here there are no necessary conditional independence statements between the variables, so that both DAGs represent all distributions. If $\mathbf{v}_1$ and $\mathbf{v}_2$ were constrained to be discrete random variables or jointly Gaussian with an arbitrary covariance matrix, then any distribution that factors according to Figure 6-4a could also factor according to Figure 6-4b

(a) (b)

Figure 6-4: Two possible connected DAGs with two variables.

and vice-versa. Both DAGs would then be scored identically by BIC.

This reasoning no longer holds true for the AcyGP model, because kernels restrict the space of possible covariances. To see this, consider a homogeneous AcyGP with two outputs. Assume for simplicity that each output m has kernel $k_m(\boldsymbol{x}_i, \boldsymbol{x}_j; l_m) = \exp(-|\boldsymbol{x}_i - \boldsymbol{x}_j|^2/(2l_m^2))$, has no noise, and all outputs are observed at all input locations $\boldsymbol{X}$. The distributions for the AcyGPs with structures in Figure 6-4 are zero mean Gaussians with covariance matrices described below.

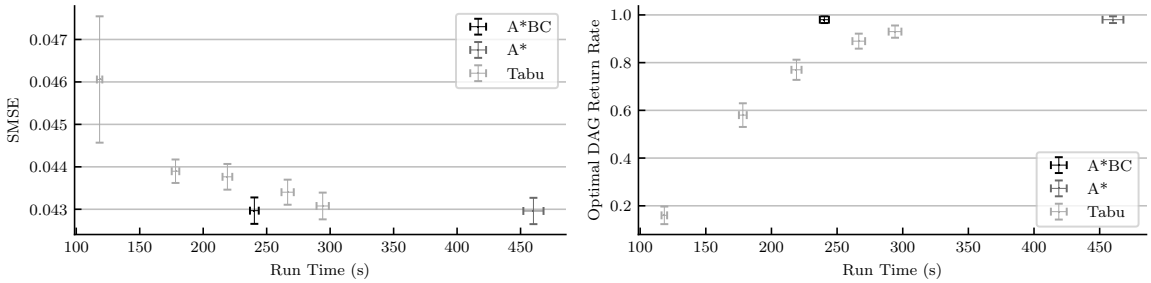
For the structure in Figure 6-4a

$$cov([\mathbf{v}_1, \mathbf{v}_2], [\mathbf{v}_1, \mathbf{v}_2]) = \begin{bmatrix} \boldsymbol{K}_1(\boldsymbol{X}, \boldsymbol{X}; l_1) & \boldsymbol{\Lambda}_{1,2}\ \boldsymbol{K}_1(\boldsymbol{X}, \boldsymbol{X}; l_1) \\ \boldsymbol{\Lambda}_{1,2}\ \boldsymbol{K}_1(\boldsymbol{X}, \boldsymbol{X}; l_1) & \boldsymbol{\Lambda}_{1,2}^2\ \boldsymbol{K}_1(\boldsymbol{X}, \boldsymbol{X}; l_1) + \boldsymbol{K}_2(\boldsymbol{X}, \boldsymbol{X}; l_2) \end{bmatrix},$$

and for the structure given in Figure 6-4b

$$cov([\mathbf{v}_1, \mathbf{v}_2], [\mathbf{v}_1, \mathbf{v}_2]) = \begin{bmatrix} \boldsymbol{K}_1(\boldsymbol{X}, \boldsymbol{X}; l_1) + \boldsymbol{\Lambda}_{2,1}^2\ \boldsymbol{K}_2(\boldsymbol{X}, \boldsymbol{X}; l_2) & \boldsymbol{\Lambda}_{2,1}\ \boldsymbol{K}_2(\boldsymbol{X}, \boldsymbol{X}; l_2) \\ \boldsymbol{\Lambda}_{2,1}\ \boldsymbol{K}_2(\boldsymbol{X}, \boldsymbol{X}; l_2) & \boldsymbol{K}_2(\boldsymbol{X}, \boldsymbol{X}; l_2) \end{bmatrix}.$$

For a sufficiently large dimensionality of $\boldsymbol{X}$, it is not always possible to select the kernel parameters and edge weights of the second covariance to equal any parameterization of the first. In particular, in the first case $cov(\mathbf{v}_1, \mathbf{v}_1)$ is parameterized by a kernel with one length scale, and in the second case it is parameterized by a kernel with two length scales, so the set of possible values of $cov(\mathbf{v}_1, \mathbf{v}_1)$ is strictly larger in the first case.



(a) (b)

Figure 6-5: Andromeda data set timing results.

6.11 Experiments

We compare the search time speedup that is gained from the use of A*BC for structural search for AcyGP problems. For the size of DAGs we consider, likelihood optimization composes 95% of structural search time. Since existing exact methods differ only in the time required for generating candidate structures, and that time is small, we only test against A* search [156] as an exact method. We also compare against locally optimal Tabu search [52], which is frequently used for structure optimization. Tabu search can be made to perform as quickly as desired by limiting the number of iterations and restarts performed, so we also compare the standardized mean squared error (SMSE) of the AcyGP under the solved structure to assess the quality of the model found, and the success rate at converging to the optimal structure. (with the lowest SMSE across all experiments)

6.11.1 Andromeda Timing Experiment

We repeat the experiment on the Andromeda data set described in Chapter 5 experiment 100 times, starting with randomly selected initial DAGs for use in structural EM. Tabu search was performed with 0 through 4 restarts and 50 iterations per restart. Results are given in Table 6.1, and plotted in Figure 6-5.

We observe a 47.8% lower training time when using A*BC compared to A* search, with no loss in predictive power or structure accuracy. A*BC performs competitively with a well-tuned Tabu search; Tabu can only achieve faster training times than A*BC

Table 6.1: Time, standardized mean squared errors, and optimal DAG convergence rate on the Andromeda timing experiment.

	A*BC	A*	Tabu				
			0 Restarts	1 Restart	2 Restarts	3 Restarts	4 Restarts
Time (sec)	240.3 ± 3.5	460.1 ± 7.9	118.4 ± 2.1	178.2 ± 3.0	219.1 ± 3.8	266.5 ± 4.8	294.1 ± 4.9
SMSE	0.04297 ± 0.00031	0.04296 ± 0.00031	0.0461 ± 0.0014	0.04390 ± 0.00028	0.04376 ± 0.00030	0.04340 ± 0.00030	0.04308 ± 0.00031
Optimal Rate	0.980 ± 0.014	0.980 ± 0.014	0.160 ± 0.037	0.580 ± 0.050	0.770 ± 0.042	0.890 ± 0.031	0.930 ± 0.026

when convergence to a suboptimal structure occurs in more than 20% of experiments, and A*BC is faster than Tabu with 3 or more restarts. The optimal number of restarts to use in Tabu would not be known prior to training, but tuning this parameter is not required when using A*BC. In this case, convergence to locally suboptimal structures increase SMSE by a few percent, but when lowest predictive errors are desired, or the structure will be examined and interpreted, A*BC provides the fastest way to reach the best known structure.

In this experiment, the number of iterations per restart in Tabu search is high enough for it to reach and become trapped in local optima, resulting in reduced rate in finding the optimal DAG and increased standardized mean squared error. Several restarts are required to reduce the probability of converging to a locally optimal structure, which eventually makes Tabu search slower than A*BC. A*BC and A* do not converge to the best known structure in all experiments because structural EM may return a locally optimal structure, depending on initial parameter guesses, even with an optimal structural search. More complex methods using restarts with multiple parameter guesses may be able to avoid this problem, but we do not explore those methods here. However, the strong performance of A*BC compared to Tabu search shows the value in performing optimal structure learning.

Number of Likelihood Evaluations

We also compare the average number of likelihoods evaluated by A*BC, A*, and Tabu search, as a measure of the efficiency of A*BC. To show the advantage of our proposed active bounding strategy and that it results in fewer evaluated likelihoods, we also compare against two alternative bound evaluation strategies.

In the 'best parents' strategy, when state s is expanded and variable $\mathbf{v}_m$ is added to form the successor, we evaluate only the optimized likelihood of the best parent set under the known bounds. That is, we evaluate $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$, where $\mathbf{v}_\Psi = \arg\max_{\mathbf{v}_\Phi \subseteq s} \overline{Score}(\mathbf{v}_m, \mathbf{v}_\Phi, \mathcal{B})$. This differs from the active bound in that computation of $\mathbf{v}_\Omega$ is ignored.

In the 'until exact' strategy, when state s is expanded and variable $\mathbf{v}_m$ is added to form the successor, we evaluate the optimized likelihood of the best parent set under the known bounds until maximum score for the successor state is known exactly. That is, we evaluate $\hat{\rho}(\mathbf{v}_m, \mathbf{v}_\Psi)$, where $\mathbf{v}_\Psi = \arg\max_{\mathbf{v}_\Phi \subseteq s} \overline{Score}(\mathbf{v}_m, \mathbf{v}_\Phi, \mathcal{B})$, and keep doing so until $\arg\max_{\mathbf{v}_\Phi \subseteq s} \overline{Score}(\mathbf{v}_m, \mathbf{v}_\Phi, \mathcal{B})$ is an element of $\mathcal{B}$.

Results are given in Table 6.2. When comparing A*BC, A*, and Tabu search, the number of evaluations agrees with the timing results. By using the active bounding strategy, search requires more than 50 fewer likelihood evaluations compared to simpler strategies, resulting in a 10.8% reduction in total number of likelihoods needed.

Table 6.2: Average number of likelihoods evaluated in the Andromeda timing experiment.

		Average Likelihoods Evaluated
A*BC	Active	447.7 ± 7.6
	Best Parents	519.5 ± 8.6
	Until Exact	501.9 ± 8.3
A*		689.3 ± 12.37
Tabu	0 Restarts	239.3 ± 4.7
	1 Restart	357.1 ± 6.3
	2 Restarts	424.0 ± 7.6
	3 Restarts	476.6 ± 8.6
	4 Restarts	513.4 ± 9.1

6.11.2 Exchange Timing Experiment

We perform 100 repeats on the Exchange data set experiment described in Chapter 5. In this experiment, the global optimum DAG is reachable through local transformations from any initial DAG with few iterations, so restarts in Tabu search are not required. We test with 5, 10, and 15 iterations. No more than 10 iterations are required by Tabu search to reach the optimal structure from any randomly generated initial structure. Results are given in Table 6.3.

Table 6.3: Time, standardized mean squared errors, and optimal DAG convergence rate on the Exchange timing experiment.

	A*BC	A*	Tabu		
			5 Iters	10 Iters	15 Iters
Time (sec)	227.3 ± 1.4	502.5 ± 0.3	170.6 ± 1.7	218.5 ± 1.8	237.1 ± 1.6
SMSE	0.4219 ± 7×10^{-9}	0.4219 ± 0.0	0.530 ± 0.018	0.4219 ± 7×10^{-9}	0.4219 ± 7×10^{-9}
Optimal Rate	1.0 ± 0.0	1.0 ± 0.0	0.360 ± 0.048	1.0 ± 0.0	1.0 ± 0.0

We find a 54.7% reduction in training time using A*BC compared to A*. Tabu with 5 iterations performs poorly in terms of SMSE and optimality rate, with 125% of the standardized mean squared error of the A*BC solution. With 10 iterations, Tabu reaches equal prediction accuracy to A*BC and is only 3.8% faster, while increasing the number of iterations to 15 causes Tabu search to be slower than A*BC. This shows that A*BC is again similar to a well-tuned Tabu search, but does not require guessing the parameters that would be needed. Note that since structural EM is not required, the complete enumeration of all likelihoods from fixed initial parameters performed by A* becomes a deterministic algorithm, causing no uncertainty in the mean SMSE.

Number of Likelihood Evaluations

We again compare the number of optimized likelihoods evaluated by A*BC, A*, and Tabu search, including different bounding strategies in A*BC. Results are given in

Table 6.4. The reduction in number of optimized likelihoods evaluated shows the advantage of A*BC over A*. However, in this experiment, there is not a significant difference in likelihoods when using the active bounding strategy versus alternatives. This may be due to the relatively low number of likelihoods needed to be evaluated in the experiment.

Table 6.4: Average number of likelihoods evaluated in the Exchange timing experiment.

		Average Likelihoods Evaluated
A*BC	Active	56.4 ± 0.4
	Best Parents	55.3 ± 0.2
	Until Exact	56.8 ± 0.3
A*		101.0 ± 0.0
Tabu	5 Iters	37.32 ± 0.06
	10 Iters	49.2 ± 0.3
	15 Iters	53.4 ± 0.4

6.12 Summary

In this chapter, we developed A* with bounding conflicts to perform score-based structure learning in problems with likelihoods that are difficult to optimize. Evaluated likelihoods are used to generate bounds on the scores of structures with unknown likelihoods, which are then used to guide search towards structures that are promising with the current information. Additional likelihoods are evaluated as search progresses, which improve the accuracy of the bounds. Experiments training AcyGP models on real, limited data problems with sparse optimal DAG structures show that A*BC can achieve training speeds that are approximately 50% faster than those achieved using A*, without a loss in solution optimality.

Chapter 7

Conclusions and Future Work

This thesis has introduced query-driven adaptive sampling, including adaptive sampling in service of queries, long duration risk-bounded planning, environment models that capture qualitative constraints from expert users, and efficient search over structure of models. In this concluding chapter, we summarize our contributions and results, and discuss possible future extensions to the work in this thesis.

7.1 Summary of Contributions and Results

This thesis was inspired by a need to apply a single adaptive sampling approach to missions with changing or novel goals. In order to realize that capability, we have introduced innovations in adaptive sampling, risk bounded planning, Gaussian process modeling, and structure learning. We now review each of these contributions.

In order to define adaptive sampling over queries, we introduced a broad query language to define problems to be solved by adaptive sampling. This query language can express the objectives solved by most adaptive sampling algorithms to date, and it provides a common interface for adaptive sampling algorithms to respond to. Under the query language, a query includes a variable of interest specified by a function that can be computed from any environment variables, a query objective, and an optional sufficient condition to be satisfied. A functional definition of variables of interest allows a user great flexibility in describing the targets of adaptive sampling.

Meanwhile, having a selection of objectives allows the goal to be clear, and relates query-driven adaptive sampling to previous adaptive sampling approaches. Sufficient conditions further draw the distinction between missions that run for a fixed length, and those that run until some level of understanding or confidence is reached. We showed that some natural queries are not suitable for adaptive sampling because they cause an agent to ignore data, and proved this does not occur in our query language.

To actually implement adaptive sampling in service of queries, we used Monte Carlo tree search based planning. Handling diverse queries required us to embed information and probability density estimators in tree search, allow early termination of rollouts, and allow consideration of only certain outcomes. In combination, these allow plans to be found for all queries in our query language, but also may be of independent interest in other applications of Monte Carlo tree search. We performed experiments simulating search for hydrocarbon seeps and identifying escape routes in a wildfire emergency. Results showed that planning with respect to a query outperforms standard maximization of information, and the fire escape scenario showed that planning can be performed for queries computed by non-trivial routines.

To allow adaptive sampling in dangerous environments, we demonstrated how to produce long duration adaptive policies that satisfy functional relationships between risk and reward. Our approach produces a series of non-adaptive plans by combining all observations at the same location in a single state. Replanning is then performed in response to new observations. By carefully constructing the risk bounds for each of the plans, we showed that the overall executed policy satisfies a risk-bounding function, and that solutions for plans could always be found after any possible observation. Through experiments with a simulated underwater vehicle, we showed that it is possible to execute risk-bounded policies with at least 20 actions. Furthermore, we showed intuitive behavior avoiding risks from tight passages as the risk bound was made tighter.

Adaptive sampling is most frequently performed in data limited environments, where pure statistical learning will frequently lead to correlations that are unjustified. In order to leverage experts' knowledge about the environment in an intuitive

manner, we developed the AcyGP model, which introduces a directed acyclic graph structure into multi-output Gaussian processes. The structure is interpretable, so that qualitative knowledge about independence and dependence relationships between variables in an environment can be modeled by selecting the structure, and relationships like monotonicity and dominance can be encoded through constraints on the edges. Structure that is not known is then solved through the use of structure learning. Our experiments on standard Gaussian process benchmarks and oceanographic data sets with limited data showed that training an AcyGP results in lower mean errors in prediction and higher likelihoods of missing data under the predicted distribution, when compared to state of the art Gaussian process regression methods.

Finally, since experts may not have complete knowledge of a structure, we considered structure learning for AcyGP models. Since it is computationally expensive to compute the likelihood of an attribute under a set of parents in a Gaussian process, we showed it is possible to significantly reduce the number of parent sets that need to be trained. We place bounds on the likelihood of an attribute given parents using likelihoods derived from larger sets of parents, and use these bounds to guide search using A*BC. We also developed heuristic rules for which likelihoods to evaluate, in order to reduce the total number needed in the course of search. Experiments showed that the time required to train an AcyGP is reduced by approximately 50% using A*BC, with no loss in structure optimality, compared to using A* search.

7.2 Extensions and Future Work

In the development of query-driven adaptive sampling, we have achieved significant generality by expressing queries through query functions, objectives, and sufficient conditions. A positive outcome of this decision is that it has allowed us to develop a planning approach that is suitable for a large class of possible queries, making our algorithm broadly applicable to many exploration problems. An obvious disadvantage to this approach is that our approach cannot be as efficient or effective as one that leverages specific assumptions or structure in the problem. When seeking to extend

research work, it is typical to propose generalizations or removal of assumptions. But since query-driven adaptive sampling is already broad, it may be more beneficial to seek more effective solutions for subsets of its capability, using more assumptions and specific structure.

Out of the extensions we propose below, 'scalable DAG structure learning' and 'more theoretically well-founded MCTS with estimators' are examples of studying a subset of the capability in this thesis while taking additional assumptions. Development of 'multi-vehicle query-driven adaptive sampling' would be an extension to the work in this thesis, while 'exact solutions for planning with risk-bounding functions' carries ideas from this thesis to additional domains.

7.2.1 Scalable Gaussian Process Structure Learning

In order to learn the structure of an AcyGP, we exploit the fact that the likelihood of a DAG can factored into the product of the likelihoods of each attribute conditioned on its parents. This allows us to determine the optimal structure by searching over combinations of parent sets, which is more efficient than training each structure individually. When data is missing, this approach requires us to resort to an expectation maximization procedure, so that the total likelihood of the structure can be predicted using estimates of the missing data.

While this procedure works well for Gaussian processes with approximately 5 to 10 attributes and on the order of 100 observations, this approach is not scalable for larger problems. The number of possible parent sets grows exponentially with the number of attributes in the environment. Even using A*BC, the number of Gaussian processes that must be trained quickly becomes too large for practical computation. A method that permits more large scale structure learning would be required for larger environments.

Recent work on structure learning, referred to as NOTEARS [158], has shown that it is possible to formulate an acyclicity constraint as a specially formulated constraint on a weighted adjacency matrix that parameterizes a DAG. The advantage of this approach is that structure can be optimized without using *any* kind of discrete space

search, simply by adding a constraint to an algorithm that optimizes DAG parameters. This means that total training time no longer scales according to the number of parent sets in a DAG, and instead grows with the number of parameters controlling the DAG, which is much more practical for large problems.

As presented, NOTEARS only allows an algorithm to solve for the single best structure, and is currently incompatible with k-best structure learning. Further research would be required to determine how to loosen this restriction. Furthermore, NOTEARS makes specific assumptions that variables are linear functions of their parents. The method would need to be generalized to be compatible with the variational optimization used in the heterogeneous AcyGP model, and to determine whether all types of variables (continuous, categorical, bounded, etc.) could still be arbitrarily combined.

7.2.2 Theoretical Guarantees on MCTS with Embedded Estimators

Query-driven adaptive sampling operates by planning using Monte Carlo tree search with embedded estimators of mutual information and probability density. As more samples are taken, values returned by the estimator form a sequence that converges to the true objective value. Within tree search, we select rollouts using the mean of elements in that sequence, which also converges to the true objective value. In this way, our tree search remains asymptotically optimal.

However, the sequence of objective values does not possess the same properties as a sequence of empirical means of samples. In particular, k-NN based density estimations and mutual information estimations *are* expectations, but they are expectations of correlated random variables, since nearest neighbor distances between independent samples are correlated. This means that the central limit theorem does not necessarily apply, and the expectation of samples is not necessarily a sub-Gaussian random variable as is assumed in the development of UCT [7, 72]. This means we lack finite time guarantees about convergence rates to optimal decisions.

In practice, the convergence properties of density and mutual information estimators are not well known, which severely limits the analysis that can be performed at this time. Recent work has shown that under specific assumptions on the distribution from which samples are drawn, k-NN entropy estimators are asymptotically Gaussian [14, 39], but bounds on the deviation from Gaussianity with finite samples have not been provided. The use of mutual information estimation within MCTS in this thesis motivates further analysis of finite-time convergence properties of information estimators, which can then be used to derive more theoretically grounded rollout selection rules.

More generally, it is known that information estimators have bounded second moments [46], and so it may be possible to use rollout selection rules based on best arm identification algorithms for distributions with heavy tails. Unfortunately, these approaches come with very weak guarantees, and we were not able to show significant improvement over UCT by using them in this thesis.

7.2.3 Multi-Vehicle Query-Driven Adaptive Sampling

The technology in this thesis was developed for a single agent that explores its environment, without the complexity of coordination with other agents. Many applications do use a single agent for exploration due to limits on cost or complexity, but it is also common to explore large environments using multiple agents. When multiple agents are used, they are most frequently deployed sequentially [23], or with each vehicle operating independently. Having multiple agents perform simultaneous cooperative operations in which information is shared is significantly more complex, but may lead to significant advantages in exploration.

Multiple exploring agents can gather information in a shorter period of time, and may carry different instrument payloads to be specialized towards a certain type of observation. Yet in order to effectively plan with multiple exploring agents, there are a number of questions that must be resolved concerning what information each agent gathers and how they share data.

When considering where each agent should explore, a simple approach is to assign

a subset of the environment to be the domain of each agent, and limit exploration to those regions. This means that agents will not explore the same locations, but there may still be some redundancy in the types of observations taken. For example, two agents may both explore areas of bleached coral at different geographic locations, but dependent on the query, more information may be obtained by having one explore bleached coral and another explore healthy coral. An alternative approach may be to have each agent explore distinct subsets of the observation space, with healthy and bleached coral being distinct observations in the previous example. This would also allow each agent to use its unique sensing capability, if the assigned parts of the observation space are determined based on instruments. An effective solution is likely to combine both approaches, so that the distances needed to be travelled to find certain types of observations are considered when assigning parts of the observation space to each agent.

Then we must handle the frequency with which each agent communicates its observations to other agents, and the number of observations each agent communicates. Intuitively, communication of observations is a slow process that would require an underwater agent to surface, and observations only need to be communicated when they would influence the behavior of another agent. One approach may be to communicate only those observations that meet some relevance criteria, such as being significantly correlated with the observations that a different agent seeks. Estimates of that correlation could be generated under each agent's current model, and when the estimate exceeds a threshold, the agent could mark the observation as needing to be communicated. The frequency of communication, in addition to the observations chosen to be communicated, may also be determined based on the degree to which belief over the other agent's observations are influenced.

7.2.4 Exact Solutions for Planning with Risk-Bounding Functions

Risk-bounding functions have been particularly useful for adaptive sampling problems, where the relationship between risk and reward is difficult to understand [8, 9]. However, the concept of a constraint on risk as a function of reward may be applicable in other domains, like motion planning and human-robot interaction. As an extreme example, it may be permissible for a robotic agent to use more risk if its actions are likely to save a human collaborator's life.

Existing solutions for constrained Markov decision processes (MDPs), such as linear programming methods [41], can be used to solve MDPs with linear risk-bounding functions. But current methods for solving problems with more general concave risk-bounding functions are not guaranteed to find optimal solutions. To generalize the application of risk-bounding functions, further research can focus on methods that do produce optimal plans.

MDPs with concave risk bounding functions can be represented as nonlinear optimization problems and solved with general purpose nonlinear constrained solvers. But solutions are likely to still be suboptimal, due to the local maxima found by nonlinear optimizers. An alternative approach may be an iterative procedure, where a local linear approximation to the risk bounding function is used, and solutions can inform the next approximation to use.

Bibliography

[1] Kareem Abdelfatah, Junshu Bao, and Gabriel Terejanu. Geospatial uncertainty modeling using stacked gaussian processes. *Environmental Modelling & Software*, 109:293–305, 2018.

[2] Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

[3] Mauricio Alvarez and Neil D Lawrence. Sparse convolved gaussian processes for multi-output regression. In *Advances in neural information processing systems*, pages 57–64, 2009.

[4] Mauricio A Álvarez and Neil D Lawrence. Computationally efficient convolved multiple output gaussian processes. *The Journal of Machine Learning Research*, 12:1459–1500, 2011.

[5] Mauricio A Álvarez, Lorenzo Rosasco, and Neil D Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4 (3):195–266, 2012.

[6] Hiromasa Arai, Crystal Maung, and Haim Schweitzer. Optimal column subset selection by a-star search. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[7] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[8] Benjamin Ayton and Brian Williams. Vulcan: a monte carlo algorithm for large chance constrained mdps with risk bounding functions. https://arxiv.org/abs/1809.01220, 2018.

[9] Benjamin Ayton, Eric Timmons, Richard Camilli, and Brian Williams. Adaptive science planning for the location of hydrocarbon seepage in the costa rica subduction zone. In *AGU Fall Meeting Abstracts*, volume 2019, pages OS23C–1793, 2019.

[10] Benjamin Ayton, Brian Williams, and Richard Camilli. Measurement maximizing adaptive sampling with risk bounding functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7511–7519, 2019.

[11] Benjamin J Ayton, Marlyse Reeves, Eric Timmons, Brian C Williams, and Michel D Ingham. Toward information-driven and risk-bounded autonomy for adaptive science and exploration. In *ASCEND 2020*, page 4149. 2020.

[12] Benjamin James Ayton. *Risk-bounded autonomous information gathering for localization of phenomena in hazardous environments*. PhD thesis, Massachusetts Institute of Technology, 2017.

[13] Mark Bartlett and James Cussens. Integer linear programming for the bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, 2017.

[14] Thomas B Berrett, Richard J Samworth, and Ming Yuan. Efficient multivariate entropy estimation via k-nearest neighbour distances. *The Annals of Statistics*, 47(1):288–318, 2019.

[15] Jonathan Binney and Gaurav S Sukhatme. Branch and bound for informative path planning. In *2012 IEEE international conference on robotics and automation*, pages 2147–2154. IEEE, 2012.

[16] Jonathan Binney, Andreas Krause, and Gaurav S Sukhatme. Informative path planning for an autonomous underwater vehicle. In *IEEE International Conference on Robotics and Automation*, pages 4791–4796, 2010.

[17] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE transactions on Robotics*, 26(3):502–517, 2010.

[18] Edwin V Bonilla, Kian M Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Advances in neural information processing systems*, pages 153–160, 2008.

[19] Phillip Boyle and Marcus Frean. Dependent gaussian processes. In *Advances in neural information processing systems*, pages 217–224, 2005.

[20] Yuheng Bu, Shaofeng Zou, Yingbin Liang, and Venugopal V Veeravalli. Estimation of kl divergence: Optimal minimax rate. *IEEE Transactions on Information Theory*, 64(4):2648–2674, 2018.

[21] Theophilos Cacoullos. Estimation of a multivariate density. *Annals of the Institute of Statistical Mathematics*, 18(1):179–189, 1966.

[22] Richard Camilli, Brian Bingham, Michael Jakuba, Hanumant Singh, and Jean Whelan. Integrating in-situ chemical sampling with auv control systems. In *Oceans' 04 MTS/IEEE Techno-Ocean'04 (IEEE Cat. No. 04CH37600)*, volume 1, pages 101–109. IEEE, 2004.

[23] Richard Camilli, Paraskevi Nomikou, Javier Escartín, Pere Ridao, Angelos Mallios, Stephanos P Kilias, and Ariadne Argyraki. The kallisti limnes, carbon dioxide-accumulating subsea pools. *Scientific reports*, 5(1):1–9, 2015.

[24] KA Campbell, JD Farmer, and D Des Marais. Ancient hydrocarbon seeps from the mesozoic convergent margin of california: carbonate geochemistry, fluids and palaeoenvironments. *Geofluids*, 2(2):63–94, 2002.

[25] Nannan Cao, Kian Hsiang Low, and John M Dolan. Multi-robot informative path planning for active sensing of environmental phenomena: a tale of two algorithms. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 7–14, 2013.

[26] Joseph E Cavanaugh and Andrew A Neath. Generalizing the derivation of the schwarz information criterion. *Communications in Statistics-Theory and Methods*, 28(1):49–66, 1999.

[27] Yuxin Chen, S Hamed Hassani, Amin Karbasi, and Andreas Krause. Sequential information maximization: When is greedy near-optimal? In *Conference on Learning Theory*, pages 338–363. PMLR, 2015.

[28] Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell, and Weiru Liu. Learning bayesian networks from data: An information-theory based approach. *Artificial intelligence*, 137(1-2):43–90, 2002.

[29] David Maxwell Chickering. A transformational characterization of equivalent bayesian network structures. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 87–98, 1995.

[30] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

[31] Shein-Chung Chow and Mark Chang. *Adaptive design methods in clinical trials*. Chapman and Hall/CRC, 2006.

[32] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[33] James Cussens, Matti Järvisalo, Janne H Korhonen, and Mark Bartlett. Bayesian network structure learning with integer programming: Polytopes, facets and complexity. *Journal of Artificial Intelligence Research*, 58:185–229, 2017.

[34] Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[35] Jnaneshwar Das, Kanna Rajany, Sergey Frolovy, Frederic Pyy, John Ryany, David A Caronz, and Gaurav S Sukhatme. Towards marine bloom trajectory prediction for auv mission planning. In *2010 IEEE International Conference on Robotics and Automation*, pages 4784–4790. IEEE, 2010.

[36] Erik A Daxberger and Bryan Kian Hsiang Low. Distributed batch gaussian process optimization. In *International Conference on Machine Learning*, pages 951–960. PMLR, 2017.

[37] Cassio P De Campos and Qiang Ji. Efficient structure learning of bayesian networks using constraints. *The Journal of Machine Learning Research*, 12: 663–689, 2011.

[38] Luis M De Campos and Juan F Huete. A new approach for learning belief networks using independence criteria. *International Journal of Approximate Reasoning*, 24(1):11–37, 2000.

[39] Sylvain Delattre and Nicolas Fournier. On the kozachenko–leonenko entropy estimator. *Journal of Statistical Planning and Inference*, 185:69–93, 2017.

[40] Eric A. DeVuyst and Paul V. Preckel. Gaussian cubature: A practitioner's guide. *Mathematical and Computer Modelling*, 45(7):787 – 794, 2007. ISSN 0895-7177. doi: https://doi.org/10.1016/j.mcm.2006.07.021. URL `http://www.sciencedirect.com/science/article/pii/S0895717706003001`.

[41] Dmitri A Dolgov and Edmund H Durfee. Stationary deterministic policies for constrained mdps with multiple rewards, costs, and discount factors. In *IJCAI*, volume 19, pages 1326–1331. Citeseer, 2005.

[42] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[43] Antonino Freno and Edmondo Trentin. *Hybrid Random Fields*, chapter 2.4. Springer, 2011.

[44] Nir Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proc. 14th International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, 1997.

[45] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[46] Weihao Gao, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Estimating mutual information for discrete-continuous mixtures. *Advances in neural information processing systems*, 30, 2017.

[47] Aurélien Garivier and Olivier Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual conference on learning theory*, pages 359–376. JMLR Workshop and Conference Proceedings, 2011.

[48] R Garnett, Y Krishnamurthy, X Xiong, J Schneider, and R Mann. Bayesian optimal active search and surveying. In *Proceedings of the International Conference on Machine Learning*. International Machine Learning Society, 2012.

[49] Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24: 81–108, 2005.

[50] Evarist Giné and Armelle Guillou. Rates of strong uniform consistency for multivariate kernel density estimators. In *Annales de l'Institut Henri Poincare (B) Probability and Statistics*, volume 38, pages 907–921. Elsevier, 2002.

[51] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[52] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[53] Gene H Golub and John H Welsch. Calculation of gauss quadrature rules. *Mathematics of computation*, 23(106):221–230, 1969.

[54] Pierre Goovaerts et al. *Geostatistics for natural resources evaluation*. Oxford University Press on Demand, 1997.

[55] Rishi Graham and Jorge Cortés. Cooperative adaptive sampling of random fields with partially known covariance. *International Journal of Robust and Nonlinear Control*, 22(5):504–534, 2012.

[56] Peter Grassberger. Entropy estimates from insufficient samplings. *arXiv preprint physics/0307138*, 2003. URL `https://arxiv.org/pdf/physics/0307138.pdf`.

[57] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[58] Evaggelos V Hatzikos, Grigorios Tsoumakas, George Tzanis, Nick Bassiliades, and Ioannis Vlahavas. An empirical study on sea water quality prediction. *Knowledge-Based Systems*, 21(6):471–478, 2008.

[59] Nils-Bastian Heidenreich, Anja Schindler, and Stefan Sperlich. Bandwidth selection for kernel density estimation: a review of fully automatic selectors. *AStA Advances in Statistical Analysis*, 97(4):403–433, 2013.

[60] Gregory Hitz, Enric Galceran, Marie-Ève Garneau, François Pomerleau, and Roland Siegwart. Adaptive continuous-space informative path planning for online environmental monitoring. *Journal of Field Robotics*, 34(8):1427–1449, 2017.

[61] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014.

[62] Jimin Hwang, Neil Bose, and Shuangshuang Fan. Auv adaptive sampling methods: A review. *Applied Sciences*, 9(15):3145, 2019.

[63] Fuwu Ji, Huaiyang Zhou, Qunhui Yang, Hang Gao, Hu Wang, and Marvin D Lilley. Geochemistry of hydrothermal vent fluids and its implications for subsurface processes at the active longqi hydrothermal field, southwest indian ridge. *Deep Sea Research Part I: Oceanographic Research Papers*, 122:41–47, 2017.

[64] Shali Jiang, Gustavo Malkomes, Matthew Abbott, Benjamin Moseley, and Roman Garnett. Efficient nonmyopic batch active search. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

[65] Shali Jiang, Benjamin Moseley, and Roman Garnett. Cost effective active search. *Advances in Neural Information Processing Systems*, 32, 2019.

[66] M Chris Jones, James S Marron, and Simon J Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American statistical association*, 91(433):401–407, 1996.

[67] Michael I Jordan. Graphical models. *Statistical science*, 19(1):140–155, 2004.

[68] Andre G Journel and Charles J Huijbregts. *Mining geostatistics*, volume 600. Academic press London, 1978.

[69] Alan G Judd. Natural seabed gas seeps as sources of atmospheric methane. *Environmental Geology*, 46(8):988–996, 2004.

[70] Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

[71] Robert Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. *Advances in Neural Information Processing Systems*, 17:697–704, 2004.

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

[73] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[74] LF Kozachenko and Nikolai N Leonenko. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16, 1987.

[75] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

[76] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, volume 7, pages 1650–1654, 2007.

[77] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(2), 2008.

[78] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[79] Jarosław Kwapień, Sylwia Gworek, Stanisław Drożdż, and Andrzej Górski. Analysis of a network structure of the foreign currency exchange market. *Journal of Economic Interaction and Coordination*, 4(1):55, 2009.

[80] Dirk P Laurie. Computation of gauss-type quadrature formulas. *Journal of Computational and Applied Mathematics*, 127(1-2):201–217, 2001.

[81] Neil D Lawrence, Guido Sanguinetti, and Magnus Rattray. Modelling transcriptional regulation using gaussian processes. In *Advances in Neural Information Processing Systems*, pages 785–792, 2007.

[82] Sangmin Lee and Seoung Bum Kim. Parallel simulated annealing with a greedy algorithm for bayesian network structure learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1157–1166, 2019.

[83] Gayle Leen, Jaakko Peltonen, and Samuel Kaski. Focused multi-task learning in a gaussian process framework. *Machine learning*, 89(1-2):157–182, 2012.

[84] Pierre FJ Lermusiaux. Adaptive modeling, adaptive data assimilation and adaptive sampling. *Physica D: Nonlinear Phenomena*, 230(1-2):172–196, 2007.

[85] Michael P Lesser, Marc Slattery, and James J Leichter. Ecology of mesophotic coral reefs. *Journal of experimental marine biology and ecology*, 375(1-2):1–8, 2009.

[86] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[87] Haitao Liu, Jianfei Cai, and Yew-Soon Ong. Remarks on multi-output gaussian process regression. *Knowledge-Based Systems*, 144:102–121, 2018.

[88] Don O Loftsgaarden and Charles P Quesenberry. A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3): 1049–1051, 1965.

[89] Damiano Lombardi and Sanjay Pant. Nonparametric k-nearest-neighbor entropy estimator. *Physical Review E*, 93(1):013310, 2016.

[90] Kian Hsiang Low, John Dolan, and Pradeep Khosla. Information-theoretic approach to efficient adaptive path planning for mobile robotic environmental sensing. In *Proceedings of the International conference on automated planning and scheduling*, volume 19, 2009.

[91] YP Mack and Murray Rosenblatt. Multivariate k-nearest neighbor density estimates. *Journal of Multivariate Analysis*, 9(1):1–15, 1979.

[92] David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

[93] Roman Marchant and Fabio Ramos. Bayesian optimisation for informative continuous path planning. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6136–6143. IEEE, 2014.

[94] Arman Melkumyan and Fabio Ramos. Multi-kernel gaussian processes. In *Twenty-second international joint conference on artificial intelligence*, 2011.

[95] J Casey Moore and Peter Vrolijk. Fluids in accretionary prisms. *Reviews of Geophysics*, 30(2):113–135, 1992.

[96] Pablo Moreno-Muñoz, Antonio Artés, and Mauricio Álvarez. Heterogeneous multi-output gaussian process prediction. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[97] National Oceanic and Atmospheric Administration. Survey H10992. https://www.ngdc.noaa.gov/nos/H10001-H12000/H10992.html, 2001.

[98] Trung V Nguyen, Edwin V Bonilla, et al. Collaborative multi-output gaussian processes. In *UAI*, pages 643–652, 2014.

[99] Masahiro Ono and Brian C Williams. An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure. In *AAAI*, pages 1376–1382, 2008.

[100] Wayne A Palsson, Tien-Shui Tsou, Greg G Bargmann, Raymond M Buckley, Jim E West, Mary Lou Mills, Yuk Wing Cheng, and Robert E Pacunski. The biology and assessment of rockfishes in puget sound. *Washington Department of Fish and Wildlife, Fish Management Division, Olympia, Washington, USA*, 2009.

[101] Shuo Pang. Plume source localization for auv based autonomous hydrothermal vent discovery. In *OCEANS 2010 MTS/IEEE SEATTLE*, pages 1–8. IEEE, 2010.

[102] Shuo Pang and Jay A Farrell. Chemical plume source localization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(5):1068–1080, 2006.

[103] Liam Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

[104] Pekka Parviainen, Hossein Shahrabi Farahani, and Jens Lagergren. Learning bounded tree-width bayesian networks using integer linear programming. In *Artificial Intelligence and Statistics*, pages 751–759. PMLR, 2014.

[105] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[106] Judea Pearl and Thomas S Verma. A theory of inferred causation. In *Studies in Logic and the Foundations of Mathematics*, volume 134, pages 789–811. Elsevier, 1995.

[107] Fernando Pérez-Cruz. Kullback-leibler divergence estimation of continuous distributions. In *2008 IEEE international symposium on information theory*, pages 1666–1670. IEEE, 2008.

[108] Oscar Pizarro. Fk180119 30-day post cruise report. Technical report, 2019. URL `https://schmidtocean.org/wp-content/uploads/FK180119-30day-Post-Cruise-Report.pdf`.

[109] Samantha A Price, R Holzman, Thomas J Near, and Peter C Wainwright. Coral reefs promote the evolution of morphological diversity and ecological novelty in labrid fishes. *Ecology letters*, 14(5):462–469, 2011.

[110] James Requeima, William Tebbutt, Wessel Bruinsma, and Richard E Turner. The gaussian process autoregressive regression model (gpar). In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1860–1869, 2019.

[111] Robert W Robinson. Counting unlabeled acyclic digraphs. In *Combinatorial mathematics V*, pages 28–43. Springer, 1977.

[112] Brian C Ross. Mutual information between discrete and continuous data sets. *PloS one*, 9(2):e87357, 2014.

[113] Lewis A Rossman. Reliability-constrained dynamic programing and randomized release rules in reservoir management. *Water Resources Research*, 13(2):247–255, 1977.

[114] Heiko Sahling, Douglas G Masson, César R Ranero, Veit Hühnerbach, Wilhelm Weinrebe, Ingo Klaucke, Dietmar Bürk, Warner Brückmann, and Erwin Suess. Fluid seepage at the continental margin offshore costa rica and southern nicaragua. *Geochemistry, Geophysics, Geosystems*, 9(5), 2008.

[115] AB Sanin, IG Mitrofanov, ML Litvak, A Malakhov, WV Boynton, G Chin, G Droege, LG Evans, J Garvin, DV Golovin, et al. Testing lunar permanently shadowed regions for water ice: Lend results from lro. *Journal of Geophysical Research: Planets*, 117(E12), 2012.

[116] Pedro Santana, Sylvie Thiébaux, and Brian Williams. Rao*: an algorithm for chance constrained pomdps. In *Proc. AAAI Conference on Artificial Intelligence*, 2016.

[117] Hannah M Sargeant, Valentin Tertius Bickel, Casey I Honniball, Sabrina N Martinez, Alexander Rogaski, Samantha K Bell, Ellen C Czaplinski, Benjamin E Farrant, Elise M Harrington, Gavin D Tolometti, et al. Using boulder tracks as a tool to understand the bearing capacity of permanently shadowed regions of the moon. *Journal of Geophysical Research: Planets*, 125(2):e2019JE006157, 2020.

[118] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

[119] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[120] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[121] Marco Scutari, Claudia Vitolo, and Allan Tucker. Learning bayesian networks from big data with greedy search: computational complexity and efficient implementation. *Statistics and Computing*, 29(5):1095–1108, 2019.

[122] Hayk Sedrakyan and Nairi Sedrakyan. *Algebraic inequalities*. Springer, 2018.

[123] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[124] Myriam Sibuet and Karine Olu. Biogeography, biodiversity and fluid dependence of deep-sea cold-seep communities at active and passive margins. *Deep Sea Research Part II: Topical Studies in Oceanography*, 45(1-3):517–567, 1998.

[125] Bernard W Silverman. *Density estimation for statistics and data analysis*. Chapman & Hall, 1986.

[126] Aarti Singh, Robert Nowak, and Parmesh Ramanathan. Active learning for adaptive mobile sensing networks. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 60–68. ACM, 2006.

[127] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[128] Eleftherios Spyromitros-Xioufis, Grigorios Tsoumakas, William Groves, and Ioannis Vlahavas. Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, 104(1):55–98, 2016.

[129] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 1015–1022, 2010.

[130] Erik B Sudderth, Martin J Wainwright, and Alan S Willsky. Embedded trees: Estimation of gaussian processes on graphs with cycles. *IEEE Transactions on Signal Processing*, 52(11):3136–3150, 2004.

[131] Ken Takai and Kentaro Nakamura. Archaeal diversity and community development in deep-sea hydrothermal vents. *Current Opinion in Microbiology*, 14 (3):282–291, 2011.

[132] YW Teh, M Seeger, and MI Jordan. Semiparametric latent factor models. In *AISTATS 2005-Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

[133] Marc Teyssier and Daphne Koller. Ordering-based search: a simple and effective algorithm for learning bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 584–590, 2005.

[134] Steven K Thompson. Adaptive sampling in behavioral surveys. *NIDA Research Monograph*, 167:296–319, 1997.

[135] Terje Thorsnes, Shyam Chand, Harald Brunstad, Aivo Lepland, and Petter Lågstad. Strategy for detection and high-resolution characterization of authigenic carbonate cold seep habitats using ships and autonomous underwater vehicles on glacially influenced terrain. *Frontiers in Marine Science*, 6:708, 2019.

[136] Eric Michael Timmons and Brian Charles Williams. Best-first enumeration based on bounding conflicts, and its application to large-scale hybrid estimation. *Journal of Artificial Intelligence Research*, 67:1–34, 2020.

[137] Fan Hsun Tseng, Tsung Ta Liang, Cho Hsuan Lee, Li Der Chou, and Han Chieh Chao. A star search algorithm for civil uav path planning with 3g communication. In *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 942–945. IEEE, 2014.

[138] Chris Tuffley and Mike Steel. Links between maximum likelihood and maximum parsimony under a simple model of site substitution. *Bulletin of mathematical biology*, 59(3):581–607, 1997.

[139] JA Vargas-Guzmán, AW Warrick, and DE Myers. Coregionalization by linear combination of nonorthogonal components. *Mathematical Geology*, 34(4):405–419, 2002.

[140] Arun Venkitaraman, Saikat Chatterjee, and Peter Handel. Gaussian processes over graphs. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5640–5644. IEEE, 2020.

[141] Jay M Ver Hoef and Ronald Paul Barry. Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Planning and Inference*, 69(2):275–294, 1998.

[142] Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pages 255–270, 1990.

[143] KL Von Damm, SE Oosting, R Kozlowski, LG Buttermore, DC Colodner, HN Edmonds, JMm Edmond, and JM Grebmeier. Evolution of east pacific rise hydrothermal vent fluids following a volcanic eruption. *Nature*, 375(6526): 47–50, 1995.

[144] Peter Vrolijk, Lori Summa, Benjamin Ayton, Paraskevi Nomikou, Andre Huepers, Frank Kinnaman, Sean Sylva, David Valentine, and Richard Camilli. Using a ladder of seeps with computer decision processes to explore for and evaluate cold seeps on the costa rica active margin. *Frontiers in Earth Science*, 9:143, 2021.

[145] Martin J Wainwright, Erik B Sudderth, and Alan S Willsky. Tree-based modeling and estimation of gaussian processes on graphs with cycles. In *Advances in neural information processing systems*, pages 661–667, 2001.

[146] Brian C Williams and Robert J Ragno. Conflict-directed a* and its role in model-based embedded systems. *Discrete Applied Mathematics*, 155(12):1562–1595, 2007.

[147] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[148] Gregory D Williams, Phillip S Levin, and Wayne A Palsson. Rockfish in puget sound: An ecological history of exploitation. *Marine Policy*, 34(5):1010–1020, 2010.

[149] Stefan B Williams, Oscar Pizarro, Ian Mahon, and Matthew Johnson-Roberson. Simultaneous localisation and mapping and dense stereoscopic seafloor reconstruction using an auv. In *Experimental robotics*, pages 407–416. Springer, 2009.

[150] Stefan B Williams, Oscar Pizarro, Michael Jakuba, and Neville Barrett. Auv benthic habitat mapping in south eastern tasmania. In *Field and Service Robotics*, pages 275–284. Springer, 2010.

[151] Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning*, pages 1067–1075, 2013.

[152] Charley M Wu, Eric Schulz, Maarten Speekenbrink, Jonathan D Nelson, and Björn Meder. Mapping the unknown: The spatially correlated multi-armed bandit. *bioRxiv*, page 106286, 2017.

[153] Xianchao Xie and Zhi Geng. A recursive method for structural learning of directed acyclic graphs. *The Journal of Machine Learning Research*, 9:459–483, 2008.

[154] Yunfei Xu and Jongeun Choi. Adaptive sampling for learning gaussian processes using mobile sensor networks. *Sensors*, 11(3):3051–3066, 2011.

[155] Namik Kemal Yilmaz, Constantinos Evangelinos, Pierre FJ Lermusiaux, and Nicholas M Patrikalakis. Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. *IEEE Journal of Oceanic Engineering*, 33(4):522–537, 2008.

[156] Changhe Yuan and Brandon Malone. Learning optimal bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48:23–65, 2013.

[157] Bin Zhang and Gaurav S Sukhatme. Adaptive sampling for estimating a scalar field using a robotic boat and a sensor network. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3673–3680. IEEE, 2007.

[158] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31, 2018.