Planning for an Uncertain Future: Tree-based Methods for Real-Time Fault Estimation, Collision Avoidance, and Multi-Agent Reconfiguration

Thesis by
James Francis Ragan III

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy in Space Engineering

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2025
Defended February 3, 2025

ACKNOWLEDGEMENTS

I am extremely grateful to the many people who have helped me reach this point of my academic career.

To my advisor, Soon-Jo Chung, for his continual support of me and my research interests throughout my Ph.D. His continued encouragement to better my research and seek out my full potential have lead me to greater success than I thought possible. I am also deeply grateful for the academic freedom he has provided me and the encouragement and support he has given along the way.

To my committee, Michael Watkins, Fred Y. Hadaegh, and Richard Murray for their guidance and feedback which have helped refine my work and my academic story.

To my mentors at the University of Washington, Carl Knolwen and Sarah Tuttle, who encouraged me to pursue an advanced degree and supported my applications.

To my many other mentors including Amir Rahmani, Joseph Mattingly, Richard Grugel, Tracie Prater, and Oliver Fraser, who helped me develop as an engineer, scientist, and teacher.

To John Fraser, who first made me believe that I belong at Caltech and that I could earn a Ph.D.

To my sponsors, the Aerospace Corporation, JPL, DARPA, and the Technology Innovation Institute, for funding and supporting my research, as well as the members of these organizations who provided valuable feedback and suggestions.

To the many people who work tirelessly to keep GALCIT running and who always made time to answer my questions and help fix any problems, including Christine Ramirez, Jamie Meighen-Sei, Liza Bradulina, and Norma Baltazar.

To the members of JPL's 347N group, who welcomed me as a co-op during the summer of 2023, including Sina Aghli who shared an algorithm which inspired a portion of my array-based MCTS implementation.

To my GALCIT cohort, without whom I would not have made it through the first year, let alone the remaining four and a half. Thursday lunches at Ernie's was a highlight of my week. I would happily work on problem sets at way too late an hour with each of you again.

To the members of the ARCL lab. My primary research collaborators, Benjamin Rivière and Joshua Ibrahim, who I am grateful for the opportunity to have worked with and learned from. Matt Anderson, Sorina Lupu, and Yashwanth Nakka, who were invaluable in teaching me about the lab's facilities and procedures and have been excellent technical advisors throughout my time here. Nikhil Ranganathan, Haeyoon Han, Michael O'Connell, Kai Matsuka, and Daniel Mitchell who have been excellent office mates and have always been willing to talk about my spur of the moment ideas. Tom Hagander, Arnauld Martinez, and Andres Torres who all exceeded my expectations as summer students and taught me how to be a better mentor. Narin Seraydarian and Paula Mark, who kept our group running. And perhaps most importantly, all of the members who have made this group the best place I can imagine to develop as an autonomy, robotics, and controls specialist.

To the members of the GALCIT Grad Student Council, who selflessly worked to make GALCIT as great of a department as it could be, and who always supported and joined my departmental games nights.

To the Caltech Y, which has always been a very welcoming place on campus for me. The members of the outdoors committee, with whom I planned many adventures. The RISE tutoring program, which let me give back to the community and grow as a teacher. And the ever welcoming staff and volunteer board, which make this unique Caltech institution possible.

To the members of KISS, including Michele Judd, Antonio Soriano, and Harriet Brettle, who created an extremely welcoming environment to grow as a member of the space community, and amazing opportunities to learn, network, and socialize.

To the many good friends I've made since moving to Pasadena, with whom I've enjoyed many board game sessions including Michael Stramenga, Alex Acosta, Branson Davis, Ying Luo, Noel Esparza-Duran, Meredith Hooper, Wesley Yu, and Quentin Chevalier, as well as the wider GALCIT community.

To my friends from Washington, who have graciously accepted the extended periods I have buried myself in my thesis and academics, and have always been supportive and ready to reconnect when I am available, including Avery Brock, Aaron Goldfogel, Alex Sells, Kenrick Chan, Austin Cassayre, Monica Cassayre, and Meera Unadkat.

Finally, to my ever supportive family, who have been with me throughout my entire journey, and to my partner Azi, who has brought out the best in me and shared in countless misadventures along the way.

ABSTRACT

Autonomous spacecraft making independent high-level decisions present the promise of dramatically increased productivity in space for both exploration and economic activity. While autonomy has seen limited use in space to date owing to a lack of flight heritage, limited computational resources, and a traditionally risk adverse industry, the growing numbers of spacecraft and increasingly ambitious missions will soon render the current ground-intensive mode of space operation untenable.

In this thesis, we develop two critical capabilities for an autonomous future in space. The first is proactive fault estimation, which seeks to rapidly and safely identify the root causes of onboard anomalies by planning sequences of test actions to gather information while probabilistically ensuring safety. The second is real-time reconfiguration to enable formations of spacecraft to respond quickly and effectively to changing environments or mission objectives.

We achieve both goals using various forms of Monte-Carlo Tree Search planning. By formalizing each capability as sequential decision-making problems, and developing algorithms well suited to information gathering, we show that our algorithms provably converge to optimal solutions while maintaining the ability to run in real-time on robotic spacecraft simulators. We present several algorithmic innovations, including marginalized filtering, sampling-based chance constraint evaluation, and an array-based implementation of Monte-Carlo Tree Search. Through and numerical simulations and hardware experiments, we demonstrate that these modifications enable our algorithms to outperform existing tree search methods and achieve better scaling across system complexity, noise, and simulation depth.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] J. Ragan, J. Ibrahim, S.-J. Chung, and F. Hadaegh, "Mitigating stealth attacks via game-theoretic switching in multi spacecraft systems," *International Astronautical Congress (Review at Acta Astronautica)*, 2024. [Online]. Available: `https://iafastro.directory/iac/paper/id/89213/summary/`,
J.R. led the conception of the project, theoretical analysis, algorithm development, simulation, and paper writing.

[2] J. Ragan, B. Rivière, F. Y. Hadaegh, and S.-J. Chung, "Online tree-based planning for active spacecraft fault estimation and collision avoidance," *Science Robotics*, vol. 9, no. 93, eadn4722, 2024. DOI: `10.1126/scirobotics.adn4722`. [Online]. Available: `https://www.science.org/doi/abs/10.1126/scirobotics.adn4722`,
J.R. co-led the conception of the project, led theoretical analysis, algorithm development, simulation, experiment validation, and paper writing.

[3] H. Tsukamoto, J. D. Ibrahim, J. Hajar, J. Ragan, S.-J. Chung, and F. Y. Hadaegh, "Robust optimal network topology switching for zero dynamics attacks," in *2024 63nd IEEE Conference on Decision and Control (CDC) (© 2024 IEEE)*, 2024. [Online]. Available: `https://arxiv.org/abs/2407.18440`,
J.R. contributed to conception of the project, problem formulation, and writing of the manuscript.

[4] J. Ragan*, B. Rivière*, and S.-J. Chung, "Bayesian active sensing for fault estimation with belief space tree search," *AIAA Scitech 2023 Forum*, 2023. DOI: `10.2514/6.2023-0874`,
J.R. co-led the conception of the project and theoretical analysis, led algorithm development, simulation, experiment validation, and paper writing. Best Student Paper Award in Guidance, Navigation, and Control.

[5] J. Lathrop, W. Cook, J. Ragan, and S.-J. Chung, "Applying monte carlo tree search for orbit selection in multi-agent inspection," *Proceedings of the 2022 AAS/AIAA Astrodynamics Specialist Conference*, 2022. [Online]. Available: `https://www.space-flight.org/docs/2022_summer/ASC22_FullProgram_Compiled.pdf`,
J.R. led the conception of the project, and mentored W.C. through the initial algorithm development and simulations.

[6] J. Ragan, R. Ahmed, K. Matsuka, I. Seker, S.-J. Chung, and M. Lavalle, "Optimizing formation flying orbit designs," *Advances in the Astronautical Sciences AAS/AIAA Spaceflight Mechanics*, vol. 176, 2021. [Online]. Available: `http://www.univelt.com/book=8507`,

J.R. led the conception of the project, algorithm development, simulation, and paper writing.

The * denotes equal contribution.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Chapter 1

INTRODUCTION

The history of robotic space exploration has seen continual increases in the level of onboard autonomy. From the first robotic scoop deployed on Surveyor 3 to Ingenuity's fully independent flights [1], [2], autonomy has led to dramatic increases in the capabilities of space missions to explore new domains and produce new scientific discoveries. Recently, with the rapid development of the commercial space industry, autonomy has also played a key role, in domains ranging from routine collision avoidance maneuvers [3] to emerging space traffic management proposals [4].

However, true high-level decision making is still in its infancy in the space domain. Autonomous processes are still mostly regulated to executing pre-sequenced, low level, and routine operations, such as regularly scheduled downlinks and charging operations [5], [6], flybys [7], and carefully planned demos [2]. When anomalies arise, spacecraft have historically been placed in safe mode, with the root cause carefully diagnosed from the ground [8]. Recent developments of higher level autonomy, such as the terrain the relative navigation used during the Mars 2020 landing of the perseverance rover [9], [10], have been the products of years of design and validation, specialized to specific missions with limited deployment. There are a number of reasons for the slow adoption of autonomy in space, including traditionally risk adverse postures when designing new missions, a lack of flight heritage, and limited onboard computing resources.

However, as the number of spacecraft continues to increase to an estimated 60,000 or more by 2030 [11], [12], this method of limited autonomy and substantial manual oversight will become intractable. Not only will the number of spacecraft demand ever increasing resources, but as low earth orbit becomes increasingly crowded, the number of interactions between spacecraft requiring time sensitive decisions will only grow. Further, as exploration missions venture further from Earth and explore new domains, less and less prior knowledge will be available [7]. Spacecraft capable of making high level decisions with potentially incomplete information will be needed, and emerging increases in onboard computational power will make higher levels of autonomy possible [13], [14].

In particular, we anticipate a demand for two fundamental autonomous capabilities. The first is a pressing need for proactive autonomous fault responses, particularly in safety-critical scenarios. Historically, fault management has been one of the first automated systems on board spacecraft, with early examples seen in the Viking and Voyager missions [15], [16]. Indeed, the ability to safely and autonomously respond when faults occur is a necessary prerequisite for any robust autonomy architecture. However, these systems are still limited to passive monitoring, with simple autonomous recovery methods [17]–[19], and they have limited ability to gather information or otherwise proactively manage the health of a spacecraft.

The second important capability we confider is real-time reconfiguration of formation flying multi-agent systems. These are groups of spacecraft working in close proximity, which can provide flexible, low cost, and robust architectures compared to traditional monolithic systems [20], [21]. However, to fully realize their adaptability, these formation need the ability to reconfigure on the fly to adjust to changing environments or operational objectives.

Both of these settings represent opportunities for novel applications of planning techniques in the space domain. We will focus on a particular class of tree-based methods known as Monte Carlo Tree Search (MCTS), developing new algorithms that are well suited to solving both of these challenges. Throughout the rest of this chapter, we briefly introduce our main problem settings, and outline the contributions of this thesis. A summary of each section is presented in Fig. 1.1.

1.1 Safe Active Fault Estimation

In time-critical settings, such as proximity operations during docking or landing, an autonomous spacecraft will need to rapidly self diagnose failures on the time scale of minutes, not hours or days. This setting precludes ground-in-the-loop interventions, and demands an onboard solution. In particular, we envision a proactive approach to fault estimation, by considering how to probe the system dynamics most effectively through test actions to gather information about the health and status of a spacecraft.

In Chapter 2, we formalize this problem setting by defining an active fault estimation problem. Using an information gathering reward, we reformulate this problem in a manner suitable for MCTS based methods, but find that existing partially observable methods fail to efficiently search over belief based rewards. To resolve this limitation, we introduce a marginal filter, which can efficiently decompose the belief over the joint state and fault space into a collection of conditional estimates which can be

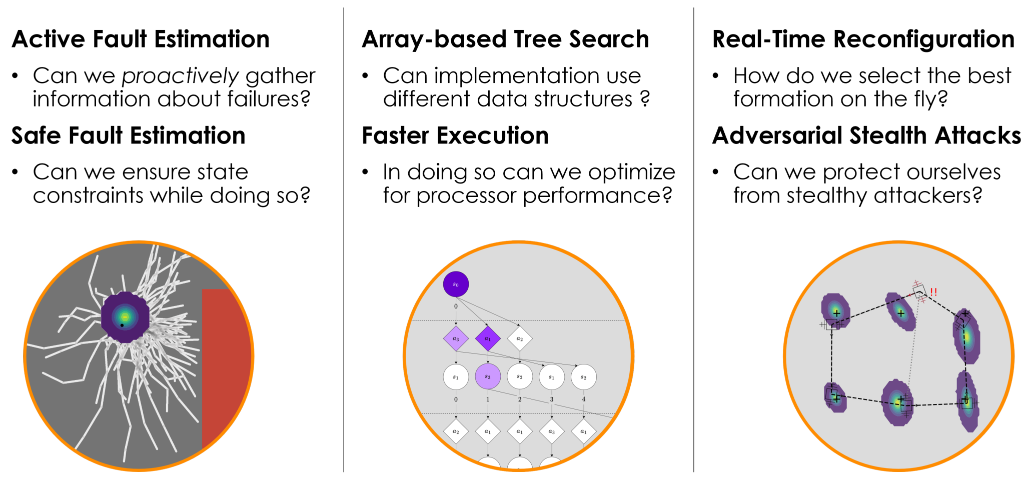

Active Fault Estimation
• Can we *proactively* gather information about failures?
Safe Fault Estimation
• Can we ensure state constraints while doing so?
Array-based Tree Search
• Can implementation use different data structures ?
Faster Execution
• In doing so can we optimize for processor performance?
Real-Time Reconfiguration
• How do we select the best formation on the fly?
Adversarial Stealth Attacks
• Can we protect ourselves from stealthy attackers?

Figure 1.1: **Contributions Overview.** We consider three core problems throughout this thesis. The first is the problem of active and safe fault estimation. We then present an array-based implementation of MCTS to optimize our processor performance. Finally, we consider extensions to multi-agent reconfiguration, in the context of stealth attacks and orbit design.

efficiently updated. This lets us introduce our algorithm, Fault Estimation via Active Sensing Tree search (FEAST), which we then demonstrate in both simulation and hardware experiments.

One potential limitation of our active sensing approach in proximity operations is the risk of inadvertently causing a collision when taking probing actions. This would be counter-productive and limits the applicability of FEAST to real world settings. To address this, in Chapter 3, we introduce the safe active fault estimation problem, which can consider chance constraints in addition to the information gathering objective from Chapter 2. We show that by further transformation of the problem, and through the use of conservative sampling based approximations, we can again reformulate into a form suitable to MCTS algorithms, creating the Safe Fault Estimation via Active Sensing Tree search (s-FEAST) algorithm. We again demonstrate its suitability for these problems in hardware and numerical experiments and compare against traditional safety methods in optimal control, demonstrating a capability gap exists when the dynamics are uncertain.

In Chapter 4, we discuss in detail why our modifications to existing partially observable tree search algorithms are necessary for the information gathering problems we consider, and provide broader context with the related fields of fault detection, isolation and recovery, active fault diagnosis, optimal control based safety, and partially observable planning. We also examine what types of problems and systems s-FEAST could be extended to beyond the motivating spacecraft example we use throughout our development. Finally, we also discuss the suitability of our algorithm to real-time problems, and our expectation that emerging computational capabilities and technologies will make our approach even more applicable.

1.2 Efficient Array-Based Tree Search

The tree search methods we consider throughout this thesis benefit significantly from increases in computational resources or efficiency, due to their asymptotic improvement with additional simulations. In the development of s-FEAST we discovered that the branching nature of MCTS algorithms limited our ability to use common accelerators such as compiled code or GPUs, especially in dynamical system where propagating the system state or updating estimators were comparably expensive. This motivated the development of a new array-based method for performing MCTS, which resolves this issue by making branching operations predictable. We introduce this algorithm in Chapter 5, and show that it outperforms traditional tree-based MCTS methods in some domains, presenting a promising direction for future development.

1.3 Real-time Multi-Agent Reconfiguration

Extending to multi-agent systems, we consider optimal planning in two settings. First, in Chapter 6 we consider the adversarial case where a system is under attack by an adversary which seeks to cause maximal disruption to the system while avoiding detection. We formulate this as a zero-sum game, and show that our tree search methods can be adapted to this two player setting to search for optimal reconfigurations and attacks. Through simulation, we demonstrate that a suitable choice of defense algorithm bounds the influence of any stealthy attacker.

Finally, in Chapter 7, we consider the problem of optimal orbit design for formation flying missions in two cases. First we design offline an optimal formation to achieve a radar science objective. We show that we can formalize the trade off between science value and mission cost, while simultaneously respecting operational constraints imposed by the radar system. We then extend to the problem of real-time orbit assignment in the context of multi-agent inspection. Using MCTS, we demonstrate the ability to rapidly reconfigure the formation of spacecraft to cooperatively inspect points of interest on a common target. Our method achieves nearly the same performance as exhaustive time-intensive brute force searches, at the cost of only a slight runtime increase compared to a previous greedy optimization approach.

Chapter 2

FAULT ESTIMATION VIA ACTIVE SENSING TREE SEARCH (FEAST)

[1] J. Ragan*, B. Rivière*, and S.-J. Chung, “Bayesian active sensing for fault estimation with belief space tree search,” *AIAA Scitech 2023 Forum*, 2023. DOI: 10.2514/6.2023-0874,

[2] J. Ragan, B. Rivière, F. Y. Hadaegh, and S.-J. Chung, “Online tree-based planning for active spacecraft fault estimation and collision avoidance,” *Science Robotics*, vol. 9, no. 93, eadn4722, 2024. DOI: 10.1126/scirobotics.adn4722. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.adn4722,

2.1 Motivation

Autonomous robots offer the potential for dramatically faster operations and better performance in domains ranging from search and rescue [22] to planetary exploration [23]. However, to achieve full autonomy, these robots must be capable of independently diagnosing and recovering from various component faults at a system level. This is especially true when the robot’s safety is a function of time-critical constraints, such as maintaining lane keeping during autonomous driving [24] or managing the accumulation of environmental degradation [25].

Spacecraft are a motivating class of autonomous systems because real-time ground-in-the-loop interventions are difficult, if not impossible, due to limited communication or large time delays. As the use of autonomous space systems increases, so does the number of failures, with 42.6% of small satellite missions between 2009 and 2016 ending in partial or complete failure [26]. On Earth, uninhabited aerial vehicles fail on the order of once every 1000 hours of operation [27], with partial failures occurring as often as every 10-50 hours in some domains [28].

In this chapter, we formalize this objective of rapid and autonomous fault identification, as a partially-observable optimal control problem and as an equivalent Partially Observable Markov Decision Process. We then present FEAST (Fault Estimation via Active Sensing Tree search), a belief-state planning algorithm that exploits a marginalization method for exact and efficient Bayesian updates. The result is an active fault-estimation method for sensor and actuator failures that maintains effective

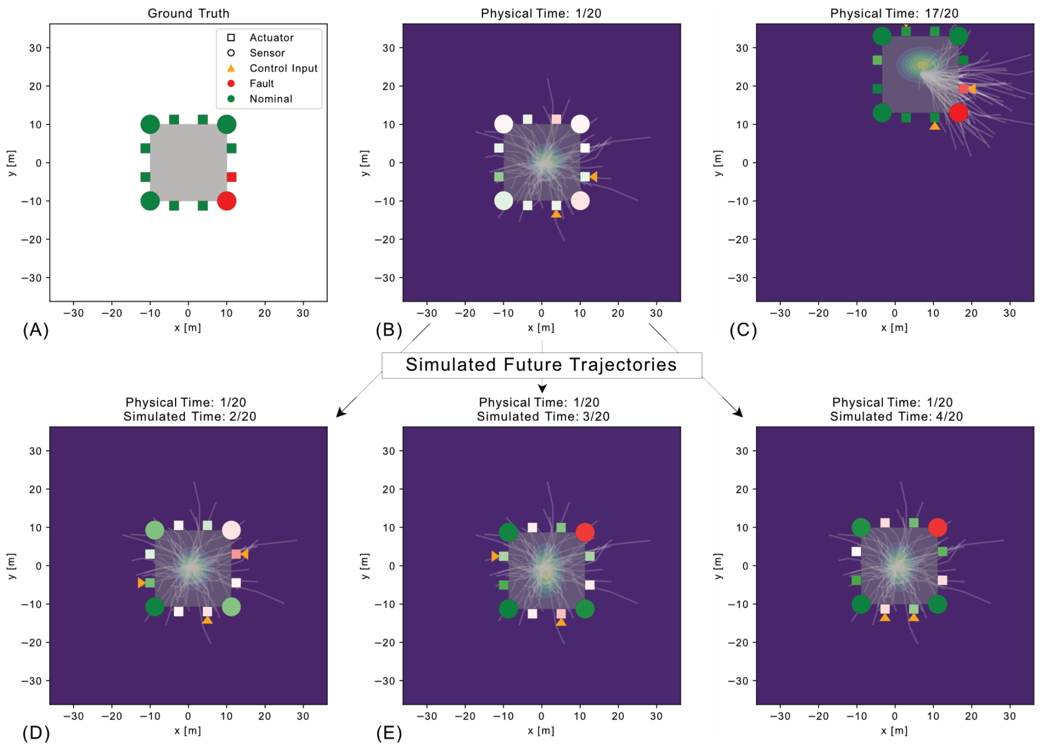


Figure 2.1: **Overview of the active fault estimation problem and method.** In (**A**), a model satellite is shown with nominal and faulty components. This ground truth is not available to the satellite. In (**B**), the experiment begins and the satellite is provided an initial belief containing little information on the status of the components, (shown by the light initial component colors) and a normal distribution on its position. The satellite must select control inputs to reveal information about the components. To choose an optimal information gathering policy, a belief-state tree is grown (mean-values shown in white) that simulates possible future scenarios, shown in (**D**), (**E**), (**F**). At the end of the experiment, (**C**), the satellite's belief has converged to the true value. We note that the size of the position belief and the satellite are scaled to improve visualization and the center of the spacecraft is the true position.

tree growth even in the presence of heavy noise. We provide theoretical guarantees of our algorithm's convergence, and validate it through numerical and hardware experiments on a spacecraft model. We demonstrate the need for an active planning solution and show that our method outperforms the state-of-the-art baseline algorithm. An overview of our problem setting and method is shown in Fig. 2.1.

2.2 Preliminaries

In this section, we define the optimal control and decision-making problems which we will use to define and transform our active fault estimation problem. We then introduce Monte Carlo Tree Search methods as a means to asymptotically solve decision-making problems.

Optimal Control

Each system we consider throughout this thesis can be modeled using general control-affine system dynamics:

$$x_k = f(x_{k-1}) + B(x_{k-1})u_k \tag{2.1}$$

where k subscript denotes a time index, $x \in X \subset \mathbb{R}^n$ is the physical state, $u \in U \subset \mathbb{R}^m$ is the control input, $f(x_k)$ is the unforced dynamics, and $B(x_k)$ is the input influence matrix.

We define the following optimal control problem, adapted from Chapter 3 of [29]:

Definition 1. *The optimal control problem is to find the sequence of controls which maximize the accumulated reward given the system in Eq.* (2.1)*, admissible control set U, and initial condition x_0:*

$$\overline{u}_K^* = \underset{\overline{u}_K \in U}{\arg\max} \sum_{k=1}^{K} R(x_{k-1}, u_k) \text{ s.t. Eq. } (2.1) \tag{2.2}$$

where $R(x_{k-1}, u_k)$ is the reward at each time step and K is the problem time horizon and the overbar notation defines a history, such as $\overline{u}_K = \{u_1, \ldots, u_K\}$.

Note that optimal control literature typically poses the optimal control problem as minimizing a cost as opposed to maximizing a reward [29]. However, the decision making problem literature typically does the opposite [30]. As we will transform the active fault estimation problem into a decision making problem, we will consider maximizing a reward for consistency throughout this thesis.

The optimal control problem given by Definition 1, describes a broad class of problems where a system can be well modeled as fully observable. Sensor noise or model uncertainty however, result in state ambiguity. To generalize to this partially observable setting, we add process noise to the system dynamics and define the noisy measurements as:

$$x_k = f(x_{k-1}) + B(x_{k-1})u_k + w_k \tag{2.3}$$

$$y_k = h(x_k) + v_k \tag{2.4}$$

where $y \in Y \subset \mathbb{R}^p$ denotes the measurement, $h(x_k)$ is the measurement equation, and the random process and measurement noise sequences w_k, v_k are assumed to be mutually independent and independent and identically distributed (i.i.d.).

In the presence of state uncertainty, it is common to write the probability distribution over the possible states as a belief, which can be updated with an observation and control input using Bayesian filtering [31]:

$$b_0(x) = \mathbb{P}(x_0) \tag{2.5}$$

$$b_k(x) = \mathbb{P}(x_k \mid \overline{y}_k, \overline{u}_k) = \frac{\mathbb{P}(y_k \mid x_k) \int \mathbb{P}(x_k \mid x_{k-1}, u_k) b_{k-1}(x) \mathrm{d}x_{k-1}}{\mathbb{P}(y_k \mid \overline{y}_{k-1}, \overline{u}_k)} \tag{2.6}$$

where $b_0(x) = \mathbb{P}(x_0)$ is the prior belief at the initial time step. The space of all possible beliefs is denoted $\mathcal{B}$.

Because the system is now stochastic, we can no longer deterministically optimize a sequence of controls. Instead, we must consider a closed loop solution, which responds to the updated belief at each time step. This concept is captured by policy functions, which are stochastic maps from belief to action, $\pi : \mathcal{B} \to U$. We denote the set of all policies as Π.

Using policies, we can define the following partially-observable optimal control problem [32]:

Definition 2. *The partially observable optimal control problem is to find the policy which maximizes the expected accumulated reward given the system in Eq.* (2.3)*, measurement in Eq.* (2.4)*, belief update in Eq.* (2.6)*, admissible control set U, initial belief b_0 and reward function R:*

$$\pi^*(b_0) = \underset{\pi \in \Pi}{\arg\max}\ \mathbb{E}\left[\sum_{k=1}^{K} R(x_{k-1}, u_k) \mid \pi, b_0\right] \quad \textit{s.t. Eqs. } (2.3), (2.4), (2.6), \quad u_k \sim \pi(b_{k-1})\ \forall k \tag{2.7}$$

where the expectation is over the stochastic policy, process and measurement noise processes.

It is often convenient to represent the expected accumulated reward given an initial belief and policy by the value function:

$$V^\pi(b_0) = \mathbb{E}\left[\sum_{k=1}^{K} R(x_{k-1}, u_k) \mid \pi, b_0\right] \quad \text{s.t. Eqs. } (2.3), (2.4), (2.6), \quad u_k \sim \pi(b_{k-1})\ \forall k \tag{2.8}$$

We will use this notation going forward, and will also denote the optimal value function corresponding to the optimal policy $\pi^*(b_0)$ as $V^*(b_0)$. Observe from Eq. (2.8) that control policies are closed-loop solutions, as desired. They select a new action at each time step in response to the belief updated via the new observation.

Despite providing a theoretical framework, the general partially observable optimal control problem is infinite-dimensional and non-convex and therefore generally intractable to solve with numerical methods. To resolve this, we reformulate the optimal control problem into a decision-making problem.

Decision Making Problems

Decision-making problems are analogous to the optimal control problem, and provide the necessary interface with the tree search methods that we will use to approximately and efficiently solve the active fault estimation problem. We adapt the convention of [30] to define the following discretized (finite horizon) Markov Decision Process (MDP):

Definition 3. *A finite horizon Markov Decision Process (MDP) is a decision-making framework defined as the collection of a set of states S, a set of actions A, state transition function T, reward function R and horizon H. At each time step t, the probability of reaching state s_k given a previous state s_{k-1} and action a_k is given by $\mathbb{P}(s_k \mid a_k, s_{k-1}) = T(s_k, a_k, s_{k-1})$, and the reward for doing so is $R(s_{k-1}, a_k)$. H defines the time horizon.*

A given MDP can be succinctly represented by the 5-tuple $\langle S, A, T, R, H\rangle$. Some literature makes distinction between stationary MDPs, such as this one, and non-stationary MDPs, where the transition and reward functions may vary with time. Similarly, generalizations can include non-deterministic rewards [30]. In this thesis however, we will consider all MDPs to be stationary with deterministic reward functions.

As a decision making problem posed over a horizon, the solution to an MDP is naturally understood in terms of a value function like that given by Eq. (2.8), with u_k being generalized to represent any actions (including control actions in dynamical systems). As was the case in the partially observable optimal control setting, for a given state in a MDP, the optimal policy, $\pi^*(s)$ returns the optimal value for that state $V^*(s) = \max_{\pi\in\Pi} V^\pi(s)$. In fact, letting $\hat{X}$ be a discretization of the continuous state space and δ the Kronecker delta function, the optimal control problem (Definition 1) and the (deterministic) MDP given by $\langle \hat{X}, U, \delta(x_t = f(x_{k-1}) + B(x_{k-1})u_k), R, K\rangle$ are equivalent in the discretization limit $\hat{X} \rightarrow X$.

To consider the partial observability introduced by measurement and process noise, we further define a (finite horizon) Partially Observable Markov Decision Process (POMDP) as:

Definition 4. *A finite horizon Partially Observable Markov Decision Process (POMDP) is a decision-making framework defined as the collection of a set of states S, a set of actions A, state transition function T, reward function R, a set of observations O, observation probability function Z, and horizon H. S, A, T, R and H are defined identically to the MDP formulation (Definition 3). In addition, the true state s_k, is hidden from the solver. Instead, an observation o_k is generated, with probability given by $\mathbb{P}(o_k|s_k, a_a) = Z(s_k, a_k, o_k)$.*

A given POMDP can be succinctly represented by the 7-tuple $\langle S, A, T, R, O, Z, H\rangle$. As in the partially observable optimal control setting, because the true state cannot be directly measured, a belief, $b(s)$, over the possible states is used. This belief is updated with each observation using the Bayesian update of Eq. (2.6), replacing x and y with s and o, respectively. The solution of the POMDP is typically understood by considering its equivalent Belief Space MDP [33]:

Definition 5. *A Belief Markov Decision Process (BMDP) is an equivalent reformulation of a POMDP (Definition 4) defined as the collection of a belief space $\mathcal{B}$, a set of actions A, belief transition function $\mathcal{T}$, belief reward function $\mathcal{R}$, and horizon H. A and H are defined identically to the MDP formulation (Definition 3). Given a finite POMDP (Definition 4) with state transition function T, reward function R, and observation probability function Z, the equivalent belief transition and belief reward functions are defined as:*

$$\mathcal{T}(b_t, a_k, b_{k-1}) = \tag{2.9}$$

$$\sum_{o \in O}\left(\mathbb{P}(b_k|b_{k-1}, a_k, o)\sum_{s_k \in S}\left(Z(s_k, a_k, o_k)\sum_{s_{k-1}\in S} T(s_k, a_k, s_{k-1})b(s_{k-1})\right)\right)$$

$$\mathcal{R}(b_{k-1}, a_k) = \sum_{s_{k-1}\in S} b(s_{k-1})R(s_{k-1}, a_k) \tag{2.10}$$

A given BMDP can be succinctly represented by the 5-tuple $\langle \mathcal{B}, A, \mathcal{T}, \mathcal{R}, H, \rangle$. Like a regular MDP, the solution to a BMDP is understood in terms a value function as:

$$V^{\pi}(b_0) = \mathbb{E}_{\pi}[\sum_{t=0}^{H} \mathcal{R}(b_k, a_k) \,|\pi, b_0], \text{ s.t. } \mathcal{T}(b_t, a_k, b_{k-1}), \; a_k \sim \pi(b_{k-1}) \quad (2.11)$$

Not coincidentally, the value function of Eq. (2.11) resembles that of Eq. (2.8). In fact, observe that when $a_k = u_k$, $\mathcal{R}(b_{k-1}, a_k) = \mathbb{E}[R(x_{k-1}, u_k)]$, and the transition dynamics are the same, the two value functions are equivalent. This reinforces the notion that a POMDP or BMDP formulation is equivalent to the partially observable optimal control formulation. We will prove this formally for our problem of interest in the next section.

We also note here that it is possible to define a BMDP that is not equivalent to an underlying POMDP by defining an alternative reward that is unstructured and directly dependent on the belief.

$$\mathcal{R}(b_{k-1}, a_k) \neq \sum_{s_{k-1} \in S} b(s_{k-1}) R(s_{k-1}, a_k) \quad (2.12)$$

We call these BMDP problems information-gathering problems, to distinguish from problems with state and action derived rewards. Instead, the objective depends on the information available to the solver itself, represented in the form of a belief. One example of such a problem is the active fault estimation problem we introduce in the next section.

Similar to the optimal control problem, exactly solving MDPs, POMDPs, or BMDPs for the optimal policy is generally intractable for high complexity problems. BMDPs introduce additional complexity because $\mathcal{B}$ is continuous for discrete state problems and infinite-dimensional for continuous state problems. However, these formulations offer one important advantage. They can be approximately solved by algorithms which asymptotically converge to the optimal value.

Asymptotic Decision Making Solvers

Instead of solving decision making problems exactly, it is possible to use numerical techniques that approximate the true optimal value function or policy. Monte Carlo Tree search (MCTS), is one such class of algorithms, providing anytime approximation of the optimal policy, by simulating future state trajectories while biasing the tree towards areas of high reward [34]. Typical theoretical analysis of MCTS provides an error bound between the root node's value estimate and the true optimal value that decreases with the number of tree simulations. The classical

Upper Confidence bounds applied to Trees (UCT) result provides value convergence at $\mathcal{O}(\log(N)/N)$ [35], while more recent results for Fixed-Depth MCTS provides value convergence at a polynomial rate $\mathcal{O}(N^{-1/2})$ [36].

MCTS has been extended to solve POMDPs via the POMCP [37] algorithm. POMCP runs UCT with a particle filter at each node, using simulated state trajectories to simultaneously estimate the optimal policy and the belief using a collection of particles. The nodes are constructed from histories of observation-action pairs where state particles are added to nodes with the same history. The resulting belief is computed as:

$$b(x) = \frac{1}{M} \sum_{i=1}^{M} \delta(x = x_i) \tag{2.13}$$

where x_i denotes the ith particle at the node for $i = 1, \ldots, M$, with M the total number of particles at the given node. POMCP argues that, at large number of samples, the belief is well approximated such that the UCT is solving the equivalent BMDP and it therefore inherits the same value convergence of UCT.

In the next section, we use the optimal control and decision-making framework to define the active fault estimation problem. In Section 2.4 we include pseudocode for POMCP and the adaptations we make with FEAST in Algorithm 2. In Chapter 4, we discuss in detail the application of POMCP to our problem setting and its limitations.

2.3 Problem Statement

In this section, we present our active fault estimation problem: to plan actions such that the resulting observations converge the belief of the underlying failure to the true fault as quickly as possible. First we define the following modification of general control-affine system dynamics given by Eqs. 2.3 and 2.4 with linear sensing:

$$x_k = f(x_{k-1}) + B(x_{k-1})(\mathbb{I} - \Phi_B)u_k + w_k \tag{2.14}$$

$$y_k = (\mathbb{I} - \Phi_C)Cx_k + v_k \tag{2.15}$$

$$u_k \in U \subseteq \{0,1\}^m \tag{2.16}$$

where $u \in U \subseteq \{0,1\}^m$ restricts the control input to a discrete set of binary m-dimensional vectors to represent thruster control, and C is the measurement matrix for linear sensing. For simplicity, we will assume the noise processes w_k and v_k are Gaussian with covariance matrices Σ_w, Σ_v, respectively, but we will later show our algorithms are not restricted to only Gaussian noise. We also note the set of control inputs U can be customized for the system of interest, for example $U \subseteq \{-1, 0, 1\}^m$ to include bi-directional actuators like reaction wheels.

The system description differs from a control-affine system only in the fault model, Φ_B, $\Phi_C = \operatorname{diag}(\phi_{B/C})$ representing changes due to failures in the actuators and sensors:

$$\phi_{B_i} = \begin{cases} 1 & \text{if } i \text{ actuator is completely failed} \\ 0 & \text{if } i \text{ actuator is nominal} \end{cases}, \qquad \phi_B = [\phi_{B_1}, \ldots, \phi_{B_m}] \quad (2.17)$$

The sensor fault model ϕ_C is defined analogously. We assume the fault state does not change with time, so we drop the time subscript k from ϕ terms and define the concatenated vector of all faults:

$$\phi_k = \phi_{k-1} = \phi = (\phi_B, \phi_C) \in \Phi \subset \{0, 1\}^{(m+p)} = \mathbb{B}^{(m+p)}, \quad |\Phi| = N_\Phi < \infty \quad (2.18)$$

where Φ is the set of N_Φ considered faults that live in the binary space of $m + p$ dimensional vectors with elements taking values of either 0 and 1. We abbreviate this space as $\mathbb{B}^{(m+p)}$. We define the augmented state by composing the physical state and fault state as $q = [x; \phi]$ where $q \in Q = X \times \Phi$.

We carry a belief over this augmented state, which we denote as $b(q)$. The initial belief is $b_0(q)$ and the Bayesian update is defined by replacing x in Eq. (2.6) with q. From this joint belief over the augmented state, we define the marginal beliefs over the failure and physical states:

$$b_k(\phi) = \int_{x \in X} b_k(x, \phi) \mathrm{d}x, \quad b_k(x) = \sum_{\phi \in \Phi} b_k(x, \phi) \quad (2.19)$$

To actively estimate faults present within the system, we are interested in maximizing the information gathered about the fault state. Traditional state and action based rewards such as in Definition 4 or Eq. (2.8) cannot efficiently capture this objective as we will discuss in detail in Chapter 4. Instead, we employ a belief based reward that maps beliefs to rewards between 0 and 1, $R : \mathcal{B} \to [0, 1]$:

$$R(b_k) = \sum_{\phi \in \Phi} (b_k(\phi))^2 \quad (2.20)$$

Note that this is an information gathering reward, as it depends directly on the belief instead of the underlying state or the action taken (Eq. (2.12)). Throughout the rest of this and the next two chapters, R and V refer to this information gathering reward and its corresponding value function:

$$V^{\pi}(b_0) = \mathbb{E}_{\pi}[\sum_{t=0}^{H} \mathcal{R}(b_k, a_k) \mid \pi, b_0], \quad \text{s.t. Eqs. (2.14), (2.15), (2.6),} \\ u_k \sim \pi(b_{k-1}) \; \forall k \tag{2.21}$$

This reward is a proxy for how confident the current belief is in the underlying fault state and has previously been proposed as an uncertainty measure [38]. Note this reward is minimized when the belief on the fault state is uniform and maximized when the belief on the fault state is a delta function. With this reward, we can now define the active fault estimation problem:

Definition 6. *The active fault estimation problem is a partially-observable optimal control problem to find the policy which maximizes the expected information gain about the fault affecting a system:*

$$\pi^*(b_0) = \arg\max_{\pi \in \Pi} \; V^{\pi}(b_0) \tag{2.22}$$

where the expectation is across the stochastic policy, measurement and process noise sequences. The corresponding optimal value is $V^*(b_0)$.

Definition 6 formalizes the active fault estimation problem we wish to solve. However, like Definition 2, this formulation is generally intractable to solve with numerical methods, leading us to seek a reformulation.

Decision Making Problem Reformulation

We reformulate the active fault estimation problem as an equivalent POMDP following Definition 4:

Lemma 1 (Equivalent POMDP Reformulation)**.** *The following POMDP is equivalent to the active fault estimation problem (Definition 6) in the discretization limit $\hat{Y} \rightarrow Y$:*

$$\begin{aligned} &\langle Q, U, \hat{Y}, R, T, Z \rangle, \\ &T(q_k, u_k, q_{k-1}) = \mathcal{N}\left(f(x_{k-1}) + B(x_{k-1})(\mathbb{I} - \Phi_B)u_k, \Sigma_w\right)(x_k) \\ &Z(q_k, u_k, y_k) \propto \mathcal{N}\left((\mathbb{I} - \Phi_C)Cx_k, \Sigma_v\right)(y_k) \end{aligned} \tag{2.23}$$

where Y, Q and U are the same observation space, augmented state space and action set as the original active fault estimation problem, R is given by Eq. (2.20),

$\mathcal{N}(\mu, \Sigma)$ represents a normal distribution with mean μ and covariance Σ, and the belief is updated between time steps following (2.6). *$\hat{Y}$ is a discretization of the observation space, Y, and Z is normalized appropriately.*

Proof. The equivalence of the two problems is shown if optimal policies of the POMDP formulation (Eq. (2.23)) are also solutions of the active fault estimation problem (Eq. (2.22)). We achieve this by showing that by construction, the two formulations are optimizing the same value function in the discretization limit.

To do so, note that the state transition function is equivalent to a deterministic transition with additive Guassian noise as given by Eq. (2.14). Similarly, in the discretization limit when $\hat{Y} = Y$, the observation probability function gives the same measurement distribution as Eq. (2.15). Since the dynamics, measurements, belief updates and rewards are the same, from Eq. (2.8) the problems must share a value function, so must be equivalent.

$\square$

In the above reformulation, assuming w_k and v_k are Gaussian gives a concise representation of T and Z. However, note that this is not necessary, so long as the distributions of w_k and v_k are known and are i.i.d., the reformulation will hold.

2.4 Methods

In the previous section, we reformulated the active fault estimation into a form that is tractable for MCTS based methods. Here, we will present the modifications we make to existing MCTS algorithms to perform better in information gathering problems. In particular, we will adapt the Partially Observable Monte Carlo Planning (POMCP) algorithm to efficiently search over belief based rewards such as Eq. (2.20) [37]. Our main innovation is to use a marginalized filter to efficiently decompose beliefs into a conditional estimate of the robot's physical state and a total estimate of the failure affecting the robot. This allows for accurate information gathering rewards within the tree search. We explain why this is necessary in Chapter 4 when discussing the limitations of existing methods.

Marginalized Filter

FEAST's marginalized filter is based on the key observation that the dynamics (Eq. (2.14)) and measurement (Eq. (2.15)) of the active sensing problem have a structure we can exploit to efficiently compute the belief update. Whereas jointly computing the belief update for the physical and fault state quickly becomes intractable, it is possible to condition on a fault then compute the conditional belief update of the physical state with a standard extended Kalman Filter (EKF) [39]. Any other nonlinear filtering approach can be used in lieu of EKF. This marginalization approach is similar to the Rao-Blackwellized filter used in FastSLAM [40], where the posterior is factored into estimations of each landmark that are conditioned on the robot path, including approaches which actively select trajectories that minimize the uncertainty of a robot's state [41]. However, instead of estimating the environment in relation to a robot, our method infers the robot's dynamics and measurement model-based on its interaction with the environment.

Our approach can be formalized in the following decomposition of the belief:

$$b_k(q) = \mathbb{P}([x_k, \phi] \mid \overline{y}_k, \overline{u}_k) = \mathbb{P}(x_k \mid \phi, \overline{y}_k, \overline{u}_k)\mathbb{P}(\phi \mid \overline{y}_k, \overline{u}_k) \tag{2.24}$$

where we use the definition of conditional probability $\mathbb{P}(A, B) = \mathbb{P}(A \mid B)\ \mathbb{P}(B)$. Consider the two terms in Eq. (2.24).

First, $\mathbb{P}(x_k \mid \phi, \overline{y}_k, \overline{u}_k)$ is the belief of a system described by Eqs. (2.14) and (2.15) with a fixed fault ϕ. Since the dynamics are known, we can directly estimate this system with an EKF or other nonlinear filter.

Second, $\mathbb{P}(\phi \mid \overline{y}_k, \overline{u}_k)$ is the belief on the failure state. Since the failure state is discrete, unchanging, and there are a finite number of failures, we can compute the Bayesian update exactly via Eq. (2.6) as:

$$\mathbb{P}(\phi \mid \overline{y}_k, \overline{u}_k) = b_k(\phi) = \frac{\mathcal{Z}_\phi(y_k, u_k, b_{k-1}(q))b_{k-1}(\phi)}{\mathcal{Z}(y_k, u_k, b_{k-1}(q))} \tag{2.25}$$

where $\mathcal{Z}_\phi$ and $\mathcal{Z}$ are conditional and unconditional measurement likelihood functions defined as:

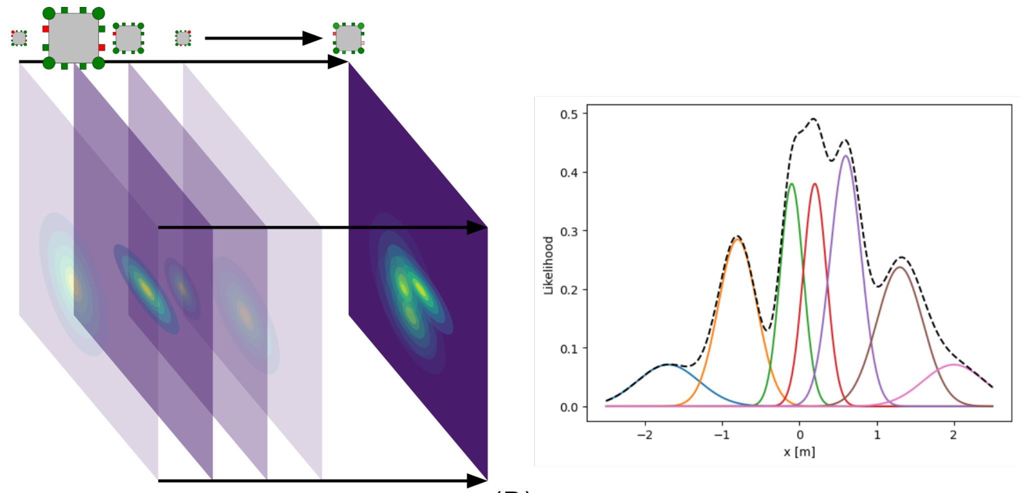
$$\mathcal{Z}_\phi(y_k, u_k, b_{k-1}(q)) = \tag{2.26}$$
$$\int_X Z\left([x_k, \phi], u_k, y_k\right) \int_X T\left([x_k, \phi], u_k, [x_{k-1}, \phi]\right) b_{k-1}(q) \mathrm{d}x_{k-1} \mathrm{d}x_k$$
$$\mathcal{Z}(y_k, u_k, b_{k-1}(q)) = \sum_{\phi \in \Phi} \mathcal{Z}_\phi(y_k, u_k, b_{k-1}(q)) \tag{2.27}$$

where the Z and T are defined in Eq. (2.23) and we split the augmented state into physical and fault state, i.e., $q = [x, \phi]$.

Combining these two distributions, we arrive at the Marginalized Filter for the active fault estimation problem: a collection Kalman filters, each corresponding to a single failure scenario, and weighted by the belief on that scenario:

$$\begin{aligned} b_k(q) &= \mathbb{P}([x_k, \phi] \mid \bar{y}_k, \bar{u}_k) \\ &= \text{EKF}_\phi[y_k, u_k, b_{k-1}(x)](x_k) \frac{\mathcal{Z}_\phi(y_k, u_k, b_{k-1}(q)) b_{k-1}(\phi)}{\mathcal{Z}(y_k, u_k, b_{k-1}(q))} \end{aligned} \tag{2.28}$$

where $\text{EKF}_\phi[y_k, u_k, b_{k-1}(x)](x_k)$ is the posterior distribution on x_k given by the Extended Kalman filter conditioned on a particular failure state ϕ. Note $\mathcal{Z}_\phi$ is also the measurement relative likelihood given by the prediction step of each conditional EKF (before measurement innovation), and that $\mathcal{Z}$ is a normalization factor, so does not need to be computed explicitly. The resulting filter is visualized in Fig. 2.2A and resembles using multiple Gaussian distributions to represent a complicated distribution as in Fig. 2.2B.



(A) (B)

Figure 2.2: **Marginalized Filter Visualization.** (**A**) Our marginalized filter combines multiple physical state estimates conditioned on different underlying faults. Here four possible spacecraft faults are shown, with the relative size indicating relative likelihood. Combining the weighted faults gives the current estimate of the failure state, and the combined physical posteriors produce a rich state estimate, much like how (**B**) combining multiple Gaussians can produce a complicated multimodal distribution.

Pseudo code for our marginalized filter is provided below. For each belief update, the filter takes in the prior composite belief over the physical and failure states, the action taken, and the measured observation. At each time step, the composite belief consists of N_Φ state estimators conditioned on each possible failure, and the probabilities of each of those failures. This allows for the belief to be decomposed and each state estimator updated individually by predicting a prior, $b_k^-(x \mid \phi_j)$, from the previous belief and action, then updated using the measured observation. This can be done with any state estimation algorithm that is suited to the system. The relative likelihood of each failure is computed from the likelihood of seeing the current observation given the predicted prior. After all estimators are updated, the relative likelihoods for each fault are normalized.

Algorithm 1: Marginalized Filter

```
1 def MF($b_{k-1}(q), u_k, y_k$):
2     $b_{k-1}(x \mid \phi), b_{k-1}(\phi) \leftarrow$ expandBelief($b_{k-1}(q)$) ;
3     for $j = 0, \ldots, N_\Phi$ do
          /* prior given fault and action (e.g.  EKF)          */
4         $b_k^-(x \mid \phi_j) \leftarrow \mathbb{P}(x_k \mid \phi_j, u_k, b_{k-1}(x \mid \phi_j))$ ;
          /* relative likelihood of fault given observation    */
5         $b_k(\phi_j) \leftarrow \mathbb{P}(y_k \mid \phi_j, b_k^-(x \mid \phi_j)) * b_{k-1}(\phi_j)$ ;
          /* posterior given prior and observation             */
6         $b_k(x \mid \phi_j) \leftarrow \mathbb{P}(x_k \mid \phi_j, y_k, b_k^-(x \mid \phi_j))$ ;
      /* normalize                                             */
7     $b_k(\phi) \leftarrow b_k(\phi)/\sum_{j=1}^{N_\Phi} b_k(\phi_j)$ ;
8 $b_k(x, \phi) \leftarrow b_k(x \mid \phi) \cdot b_k(\phi)$ ;
9 return $b_k(x, \phi)$ ;
```

We note the conditional physical state estimator can be replaced by any estimator parameterized by the failure state, including estimators for non-Gaussian processes. In particular, as the estimation propagation is the primary computational burden in the tree search, our method will benefit substantially from reusing any efficient estimators that may already exist for a system, as opposed to approaches attempting to estimate the joint physical and fault state directly. For example, one strategy to amortize real-time computation cost is to train a neural network based filter from offline data [42]. Another strategy is to perform an additional marginalization step on any states of the system that do not depend on the considered faults. This will particularly be useful to scale FEAST to high-dimensional systems with isolated faults, as only a subset of the estimation needs to be repeated for each considered fault.

FEAST Algorithm

Here we present the FEAST algorithm and discuss the changes with respect to POMCP. While POMCP uses particle filters to simultaneously estimate the belief and the optimal policy [37], FEAST uses our marginalized filter to immediately estimate the correct belief by computing the exact Bayesian update. Furthermore, we use the updated belief to compute exact rewards and generate a value estimate.We visualize the growth of our algorithm in Fig. 2.3, and include the pseudocode in Algorithm 2, with differences between FEAST and POMCP highlighted.

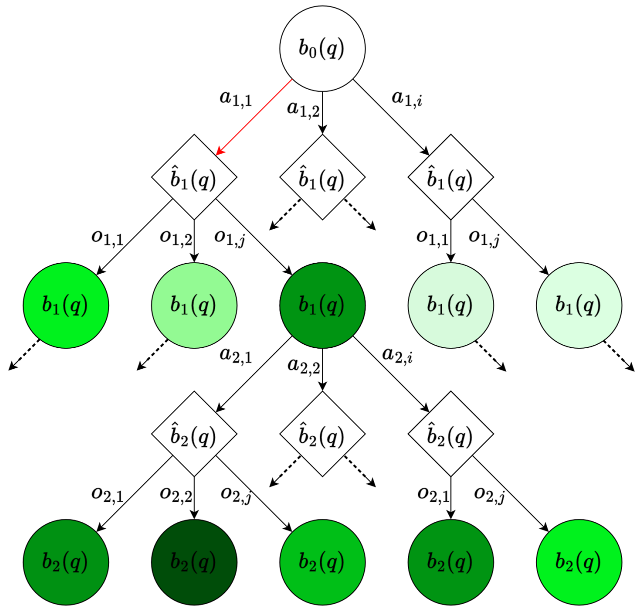


Figure 2.3: **Diagram of a tree search applied to a BMDP.** Starting from the root node and initial belief, $b_0(q)$, an action is selected, and the system propagated in simulation to create a prior belief for the next time step, $\hat{b}_1(q)$. The measurement is also simulated, and the prior belief updated accordingly. This process is repeated down the tree to the desired depth, and the resulting rewards are propagated upwards through the tree at a discounted rate. The tree growth is biased towards nodes leading to better rewards (represented here as darker shading) and the best action is returned, here $a_{1,1}$.

Both algorithms, approximate the optimal policy as a tree of nodes. A node is defined as an ordered history H of actions and observations, with corresponding number of visits $N(H)$, value estimate $\hat{V}(H)$ and belief $b(H)$. Each simulation is performed from the root node until the depth, d exceeds the maximum depth K. New states and observations are simulated by the model-based generator G. When a previously unexplored history is encountered, the simulation rolls out to the max depth by uniformly sampling random actions from the action space U. While

Algorithm 2: The POMCP and FEAST algorithms for belief-space planning. For this pseudocode, we adapt the original POMCP algorithm to our notation, with modifications made to create FEAST highlighted in blue [37]. MF refers to our marginalized filter (Algorithm 1).

```
globals: $\hat{V}(\cdot) \leftarrow 0,\ N(\cdot) \leftarrow 0$
1  def search($b_0$):
2      for $i \leftarrow 1$ to $N$ do
3          simulate($q \sim b_0, \emptyset, 0, b_0$) ;
4      return $\arg\max_u \hat{V}(u)$ ;
5  def simulate($q_d, H_d, d, b(H_d)$):
6      if $d > K$ then
7          return 0 ;
8      $u_{d+1} \leftarrow \arg\max_u \hat{V}(H_d \cup u) + c\sqrt{\frac{\log N(H_d)}{N(H_d \cup u)}}$;
9      $(q_{d+1}, y_{d+1}) \sim G(q_d, u_{d+1})$ ;
10     $H_{d+1} \leftarrow H_d \cup \{u_{d+1}, y_{d+1}\}$ ;
11     if FEAST then
12         $b(H_{d+1}) \leftarrow \mathrm{MF}(b(H_d), u_{d+1}, y_{d+1})$ ;
13     else
14         $b(H_{d+1}) \leftarrow b(H_{d+1}) \cup q_{d+1}$ ;
15     $r \leftarrow R(b(H_{d+1}))$;
16     $r \leftarrow r + \gamma$ simulate($q_{d+1}, H_{d+1}, d+1, b(H_{d+1})$);
17     if $N(H_d \cup u_{d+1}) = 0$ and $N(H_{d-1} \cup u_d) = 0$ and not FEAST then
18         return $r$
19     $N(H_d) \leftarrow N(H_d) + 1$ ;
20     $N(H_d \cup u_{d+1}) \leftarrow N(H_d \cup u_{d+1}) + 1$ ;
21     $\hat{V}(H_d \cup u_{d+1}) \leftarrow \hat{V}(H_d \cup u) + \frac{r - \hat{V}(H_d \cup u_{d+1})}{N(H_d \cup u_{d+1})}$ ;
22     return $r$ ;
```

POMCP discards nodes encountered beneath the first unexplored action (represented by $N(H_d \cup u_{d+1}) = 0$ ***and*** $N(H_{d-1} \cup u_d) = 0$)), because computing the marginalized filter is relatively expensive, FEAST saves the nodes from the rollout instead of discarding them. In practice, this tree growth is similar to the fixed depth Monte Carlo tree search proposed by Shah [36]. After completing all N simulations (or timing out), the action with the highest value estimate is returned and applied to the system. The resulting observation is used to update the system's belief, and a new tree is planned from this new root node to select the next action.

When initialized in each experiment, FEAST was given knowledge of the system dynamics through Eqs. (2.14) and (2.15), including a nominal noise model and the possible failures. A uniform initial probability over all possible failures was assumed. If prior knowledge of the relative likelihoods of each failure exists, it can be incorporated, and FEAST can in fact converge from any initial belief that does not preemptively eliminate the true failure. Overly conservative noise models can also be provided when the true noise level or other aspects of the system dynamics are uncertain. However this means that each observation will be less informative, and FEAST will take longer to converge.

Finally, we formalize the correctness our algorithm, showing that FEAST converges to the optimal solution to the active fault estimation problem.

Theorem 1 (Optimality of FEAST)**.** *Let $\mu(b_0)$ denote the policy produced by FEAST, and $\pi^*(b_0)$ denote an optimal policy to the active fault estimation problem (Definition 6). In the discretization limit $\hat{Y} \to Y$:*

$$\lim_{N\to\infty} (V^{\mu}(b_0) - V^*(b_0)) \to 0 \tag{2.29}$$

with convergence rate $O(\log N/N)$.

Proof. From Lemma 1, Definition 6 can be equivalently reformulated as a POMDP. To solve the equivalent POMDP, FEAST employs the marginalized filter given by Eq. (2.28) to perform an Bayesian update when creating a new node in the tree search, and incurs an accurate reward. Therefore, we are performing Partially Observable Upper Confidence bound applied to Trees (PO-UCT) from [37] and inherit the convergence rate from their Theorem 1.

□

2.5 Numerical Simulations

To validate FEAST empirically, we construct a series of numerical experiments and compare the performance of our method versus several baselines. We first consider a 1D single integrator system to demonstrate the ability of our method and our baselines to correctly identify the faulty components in a simple problem. To demonstrate the superior scalability of our method, we then consider a 2D double integrator system with twice as many actuators and sensors. In both systems, we also scale the noise present in the system, to demonstrate FEAST's robustness to high noise, even in complicated problems. Finally, we extend our method to a nonlinear planar spacecraft, to validate FEAST's applicability to the robotic system we consider in the next section.

Simulation Overview

In each of our simulations, we consider various spacecraft models initialized at the origin with zero velocity. We limit the faults considered to have at most 3 simultaneous actuator and sensor failures and further limit the failure space to a maximum of 42 randomly selected possibilities. This is done ensure the increase in problem difficulty comes from the dimensionality and the noise, and not the number of scenarios considered. To initialize b_0, we assume a uniform prior over the failure space. The physical belief is centered at the origin with a diagonal covariance matrix of σ_0.

To evaluate the performance of each algorithm, we use the following diagnostic reward. The primary component is the confidence of the fault state belief at each time step, which is computed by evaluating Eq. (2.20). The faster this reward increases, the more quickly an algorithm converges to a diagnosis. When the confidence in a particular fault scenario (i.e., specific combination of actuator and sensor faults) crosses a specified threshold, the algorithm terminates and returns its diagnosis. If the failure diagnosis matches the true failure, the experiment is considered a success, and otherwise, it is considered a failure. We ultimately use the product of the confidence reward and diagnostic success rate as the single metric of an algorithm's fault estimation performance because this metric rewards algorithms that rapidly converge to a high confidence while penalizing incorrect diagnoses.

Implementation details as well as specifics of each system we consider are provided in Appendix A.

Overview of baselines

We compare FEAST's ability to quickly and accurately diagnose failures against two baselines. To examine whether a passive solution is sufficient, the first is a random policy that selects actions uniformly from the admissible control set U and does not make any optimization or planning to improve the belief. Instead, the true belief is passively discovered by the estimator alone as these random actions generate observations.

To compare with another active and planned method, the other baseline we consider is the classical POMCP method adapted to belief-state planning [37]. The key difference between POMCP and FEAST is that the POMCP uses a particle filter to propagate its simulated belief during the tree search, whereas FEAST is an extension of POMCP to belief-space planning using a marginalized filter to perform exact Bayesian updates. The performance difference between our method and the POMCP baseline demonstrates the importance of high quality belief updates for efficient tree growth and better predictions of each action's information gain. In Chapter 4, we provide theoretical justification for why this modification leads to better performance in this problem setting. Between each experiment time step, we use the same marginalized filter to update the belief estimate regardless of the policy used. This results in us comparing only the quality of the actions selected by each method.

FEAST in Linear Systems

We first consider a linear, 1 degree of freedom (DOF), single integrator system as a simple example to build intuition, establish the baseline methods, and demonstrate the superior performance of active vs passive methods for the active fault estimation problem. In our validations with this system, we deploy the baseline and FEAST method for 1000 trials and plot the performance metric vs experiment time. We vary the number of simulations in the tree from $N = 15, 50, 100, 200$ and we run this experiment with shared noise parameters of $\sigma = 0.1$ and $\sigma = 0.4$ for both process and sensing noise.

We then consider the 2 DOF double integrator to demonstrate (i) the performance gap between planned vs greedy methods and (ii) the performance gap between POMCP and FEAST. We again vary the number of simulations in the tree from $N = 15, 50, 100, 200$ and we run this experiment with $\sigma = 0.4$ and $\sigma = 1.0$.

The results of both the 1 DOF and 2 DOF system is shown in Fig. 2.4. In each experiment, σ is the shared standard deviation of the process and dynamics noise, in meters, and N is the number of simulations each variant of the FEAST and POMCP algorithms performs before selecting an action, with increasing levels of N indicating more planning. In all experiments, the faster our diagnostic performance metric increases, the more rapidly and more accurately the algorithm is identifying the underlying fault. In all experiments, FEAST is diagnosing between binary sensing and actuation faults, where up to three components are completely failed.

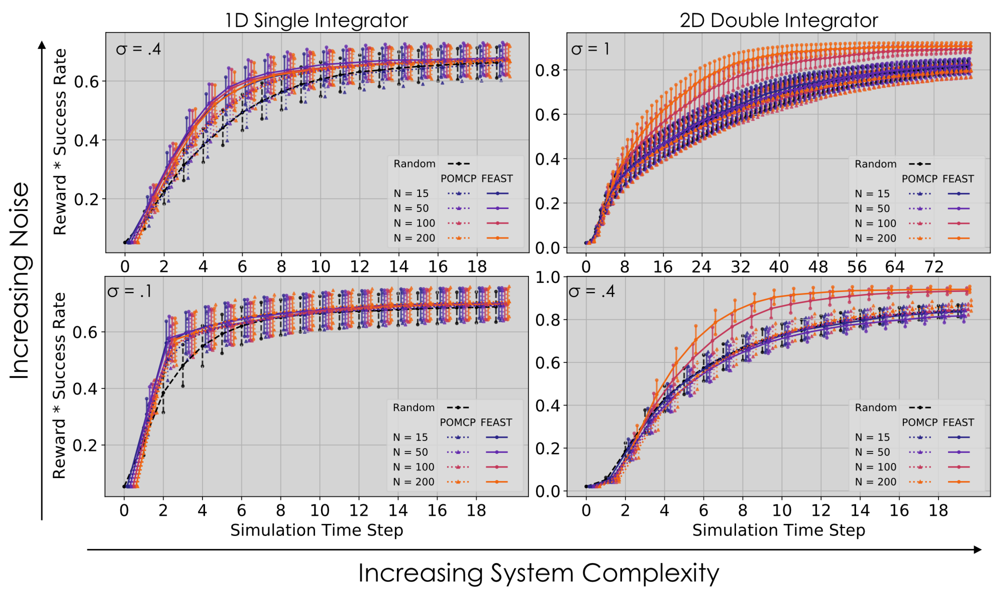
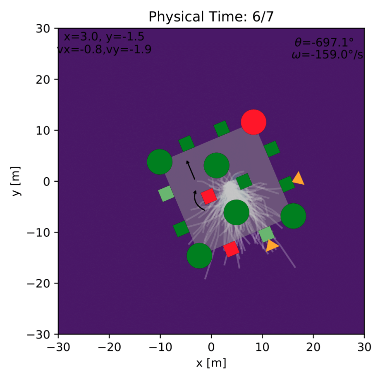
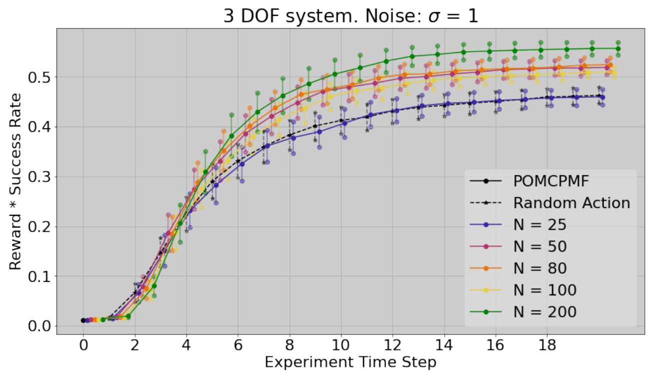


Figure 2.4: **Validation of FEAST.** The numerical performance of FEAST and POMCP across systems of increasing complexity and noise as the number of simulations per selected action, N, varies. Note the top right experiment has a longer time length. For each experiment, the performance metric vs simulation time for FEAST and its baselines is shown, with each data point averaged over 1000 randomly selected underlying faults. Note that for readability, the data for each time step is artificially spread out horizontally, and the error bars for each point are one quarter a standard deviation.

In the 1 DOF low noise scenario ($\sigma = 0.1$) in Fig. 2.4, we find that proposed active methods all outperform the passive random baseline by about 20% at the 3rd timestep, thus validating the use of an active sensing approach for rapid fault estimation. In the higher noise 1-DOF scenario ($\sigma = 0.4$) in Fig. 2.4, we find this performance gap grows, indicating that actively exciting useful observations becomes more crucial in systems with higher noise. In these 1-DOF experiments, there is a minimal performance gap between FEAST and classical POMCP methods, validating the baseline in simple systems. The exception is the POMCP variant with $N = 15$, which performs nearly identically to the passive baseline when the noise is increased. Due to the higher noise present in the system, the simulated particles that POMCP uses to update its belief have high variance and large belief error, resulting in large reward error. We show in Chapter 4 that this issue results in a breadth first search until there are sufficient simulations for the belief estimate to converge, yielding similar behavior and performance to the random policy as seen here.



(A) The 3DOF planar satellite model's orientation is the additional degree of freedom compared to the 2DOF double integrator. This also makes the system nonlinear.



(B) The performance metric vs experiment time for the proposed method and baseline, with each data point averaged over 300 initial conditions with high noise, i.e., $\sigma = 1.0$. Note the data for each time step is artificially spread out horizontally for readability, and the error bars for each point are 0.25 σ.

Figure 2.5: **Performance on a 3DOF Planar Satellite system.**

Extending to the 2 DOF planar spacecraft, in the low noise ($\sigma = 0.4$) scenario in Fig. 2.4, we find that each POMCP method collapses to the same performance as the passive baseline. This is due to the same scaling issues discussed above, and empirically validates the need for high quality belief updates within the simulated tree to explore informative actions. There is also a slight performance gap between FEAST methods as planning increases, e.g., $N = 200$ converges to a 0.9 performance metric 1.28 times faster than the $N = 100$ solution. This gap grows in the higher noise ($\sigma = 1$) scenario, suggesting that the performance gap will continue to grow in problem complexity, and demonstrating the FEAST is able to significantly outperform passive methods in high noise scenarios (more than a factor of 2 higher performance).

FEAST in a Planar Satellite System

Next we consider a 3 DOF planar satellite system to demonstrate that (i) FEAST is naturally extended to nonlinear systems when a Bayesian estimator is known and (ii) the performance gap between FEAST and passive sensing persists in complex systems. Finally, because the 3 DOF planar satellite system can be used as a model for the M-STAR spacecraft simulator hardware shown in Fig. 2.6 [43], numerical experiments on this system are a useful intermediate step for hardware demonstrations.

We present our results for the 3 DOF planar satellite system in Fig. 2.5. In this experiment we deploy the FEAST method alone for 300 trials and plot the performance metric vs experiment time. As the baseline POMCP algorithm's performance was indistinguishable from the random policy in the high noise 2 DOF experiment (Fig. 2.4), it is left out of this experiment. We vary the number of simulations in the tree from $N = 25, 50, 80, 100, 200$, and we run this experiment in the most challenging case considered earlier, with $\sigma = 1.0$. Because of the increased size of the action space, we now limit the actions considered to a random subset of 20 actions, shared across all trials. Extending the trend of previous experiments to a realistic hardware model, Fig. 2.5 demonstrates that our algorithm results in a significant improvement over passive methods.

2.6 Robotic Spacecraft Simulator Hardware Experiments

We implemented FEAST on a Multi-Spacecraft Testbed for Autonomy Research (M-STAR) robot [43], [44] using the Caltech Autonomous Robotics and Control Lab's spacecraft simulator facility, shown in Fig. 2.6. The M-STAR robot is actuated using thrusters and uses air bearings to float on a high precision flat floor, creating a very low friction environment which simulates spacecraft dynamics. A motion capture system provided position and orientation measurements and faults and noise were artificially added according to our observation model, shown in Fig. 2.7A.

To deploy FEAST in the real-time setting, we implemented FEAST in a receding horizon fashion, meaning that the planner recomputed a policy every time step, and applied only the first action to the physical system. Since the dynamics continued to propagate during the planning computation time, the state of the system when the planner began solving, x_k was different from the state when the selected action was taken, $x_{k+\delta_t}$, where δ_t was the propagation time. To synchronize these two states, we ran the same FEAST algorithm, except we planned the next action to take from the expected result of the current action. The modified tree topology is visualized in Fig. 2.7B. Instead of specifying a number of simulations to run, we took advantage of FEAST's ability to provide an anytime solution by simulating until the computation budget 0.7 seconds was exhausted and returning the best action. The selected action was then applied onboard the robot and the current observation was used by FEAST to compute the next action to take while the system dynamics propagated.

With this modification, FEAST is able to successfully identify the true failure state, demonstrating that our algorithm works on a physical nonlinear system. Images

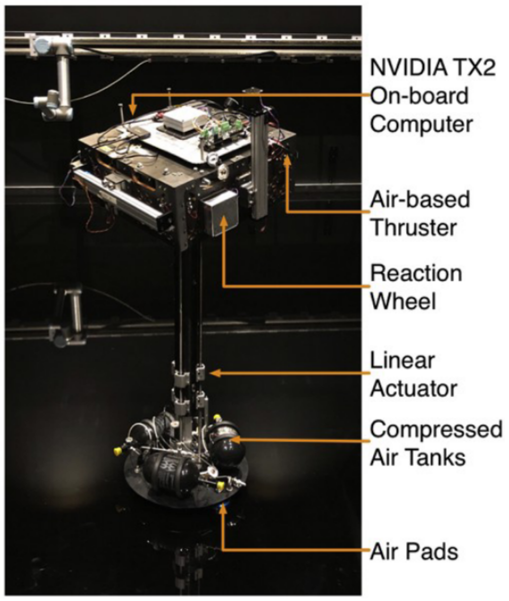


Figure 2.6: **The Caltech Autonomous Robotics and Control Lab's Spacecraft Simulator Facility and M-STAR robot.** Reproduced with permission from [44].

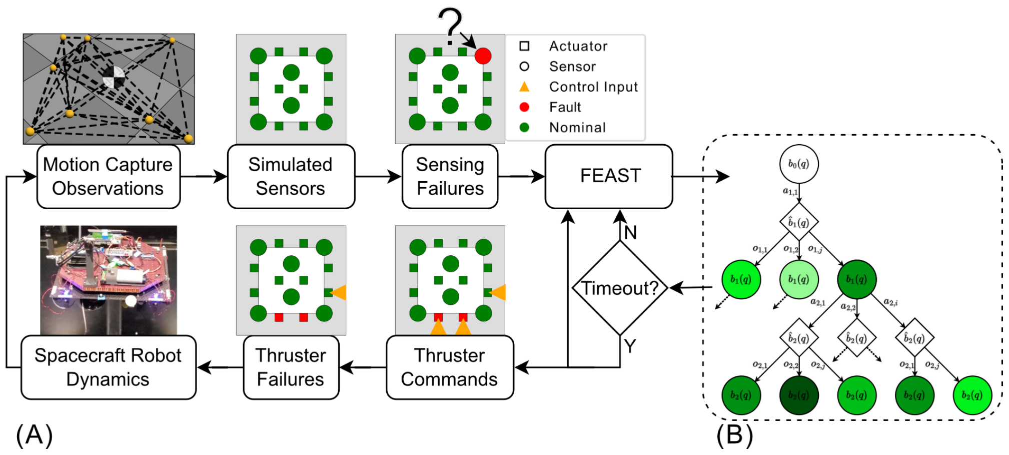
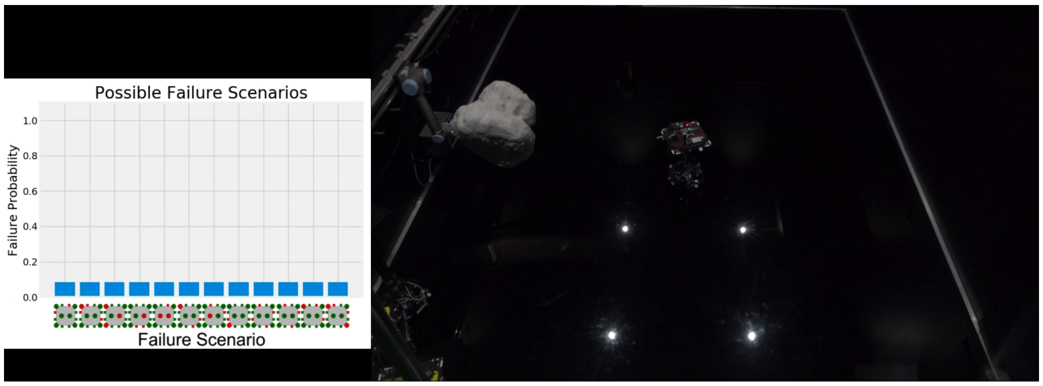
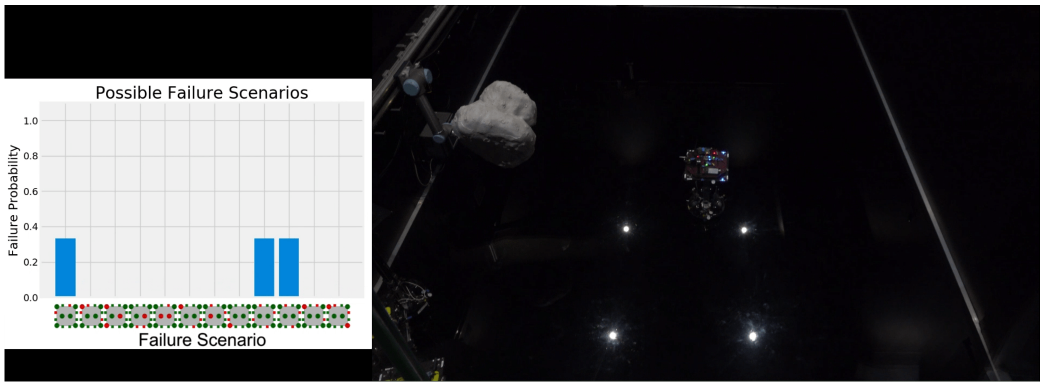


Figure 2.7: **FEAST: Real-time implementation.** (**A**) Diagram of our real-time deployment on the robotic spacecraft simulator. The FEAST block runs until the specified computation time budget is exceeded, then the best available action is returned. (**B**) Real-time FEAST is run with the first action fixed. This is the currently active action selected at timeout by the previous iteration.

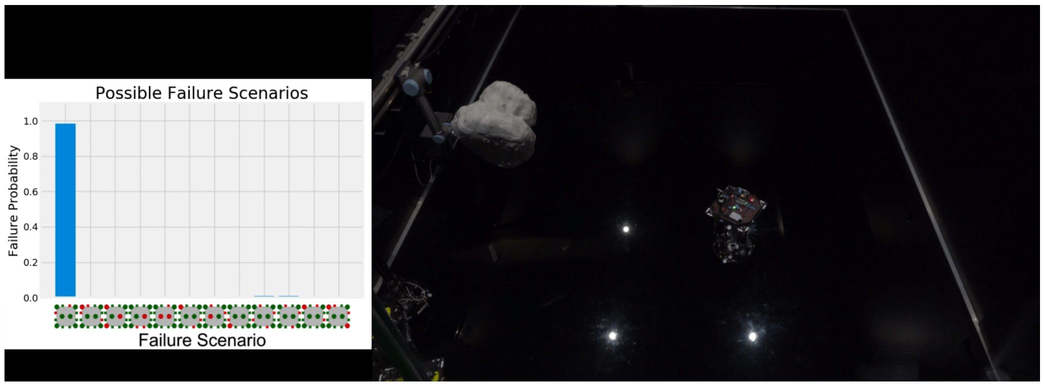
taken from a successful hardware experiment at three different time steps alongside the current belief over each considered failure scenario are shown in Fig. 2.8.



(A) t=0 seconds



(B) t=3 seconds



(C) t=7 seconds

Figure 2.8: **Robotic spacecraft simulator validation of FEAST.** Each failure scenario considered is represented pictorially, with green representing nominal behavior, and red faulty components. Thrusters are represented as squares, sensors as circles. The thrusters are illuminated blue when firing, such as in (**B**).

2.7 Chapter Summary

In this chapter, we have formalized the active fault estimation problem, and developed an algorithm to solve it. We have proven that FEAST asymptotically converges to the optimal solution, and scales better than existing partially observable planning methods across a range of system complexity and noise levels. We have also shown that FEAST can be deployed to robotic systems and successfully identify faults in real-time.

One question that naturally arises from our proposed active fault estimation scheme: how do we ensure that the actions we take to diagnose the fault, do not adversely affect the system? Returning to our motivating example in Fig. 2.1, what would happen if we were in close proximity to other spacecraft or bodies? In this case, some test actions which might help diagnose the fault state could be putting us on a collision course. Can we consider the operating environment of our system while planning for diagnostic actions? We will address these questions in the next chapter, where we formalize these desires as the safe active fault estimation problem, and extend FEAST to satisfy these safety constraints with high probability.

Chapter 3

SAFE FAULT ESTIMATION VIA ACTIVE SENSING TREE SEARCH (S-FEAST)

[1] J. Ragan*, B. Rivière*, and S.-J. Chung, "Bayesian active sensing for fault estimation with belief space tree search," *AIAA Scitech 2023 Forum*, 2023. DOI: `10.2514/6.2023-0874`,

[2] J. Ragan, B. Rivière, F. Y. Hadaegh, and S.-J. Chung, "Online tree-based planning for active spacecraft fault estimation and collision avoidance," *Science Robotics*, vol. 9, no. 93, eadn4722, 2024. DOI: `10.1126/scirobotics.adn4722`. [Online]. Available: `https://www.science.org/doi/abs/10.1126/scirobotics.adn4722`,

3.1 Motivation

Building on the active fault estimation framework developed in the previous chapter, we now consider the problem of fault estimation onboard robotic spacecraft that will soon violate state safety constraints. One such example is shown in Fig. 3.1 (B and C). Here, a robot approaches a model comet, and component failure could jeopardize mission success. In this scenario, we envision a system-level emergency response in which safe and autonomous identification of the underlying fault as quickly as possible supersedes the primary objectives of the mission. To this end, we develop s-FEAST (Safe Fault Estimation via Active Sensing Tree search), an extension of FEAST that selects diagnostic actions to gather informative observations while satisfying probabilistic state constraints at each planning step. As shown in Fig. 3.1B, the autonomous spacecraft is subject to a failure of both of its retro-thrusters. Conventional model-based passive fault detection approaches will likely not detect this failure until the spacecraft attempts to maneuver and a discrepancy between the predicted and observed states is noticed. At this point, it may be too late to maintain the safety constraints on the spacecraft's state. Similarly, methods of representing the safety of the spacecraft that are unable to consider uncertainty in the system model will not properly capture the risk of this adversarial fault. Instead, we consider actively gathering information about the fault to be a top priority, and necessary to avoid over-confident predictions of safety. With our approach, the robot proactively re-orients and diagnoses the failure, avoiding collision [see Fig. 3.1 (E and F)].

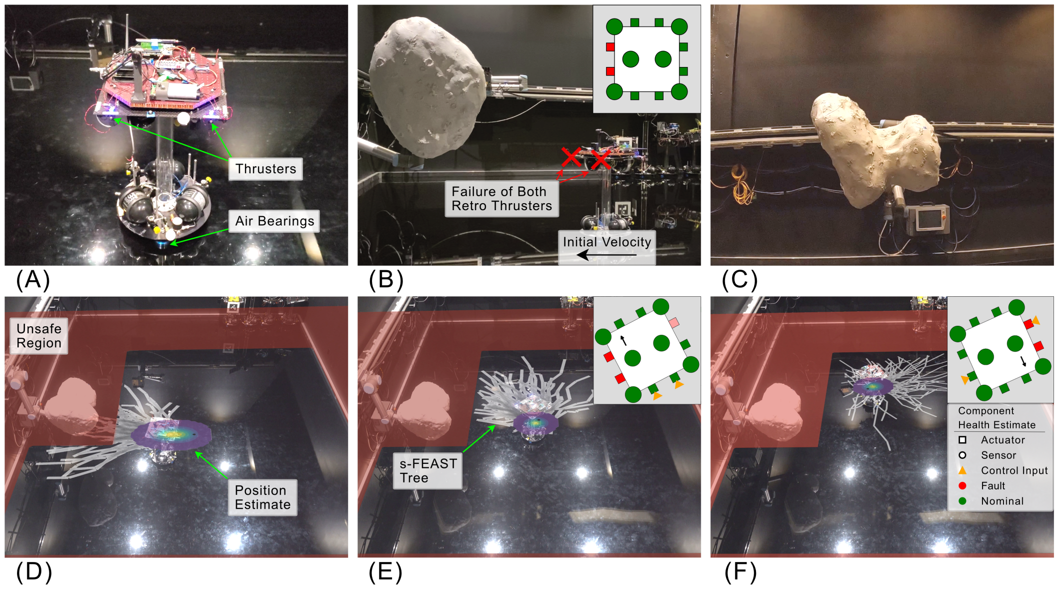


Figure 3.1: **Safe fault estimation on robotic spacecraft.** (**A**) We demonstrate our method on the Caltech Autonomous Robotics and Control Lab's Spacecraft Simulator, which creates a near frictionless environment by using graphite air bearings to create a cushion of air between the robot and the flat floor. (**B** and **C**) Both of the spacecraft robot's retro thrusters have failed, and it starts on a collision course with the comet. With no evasive actions, a collision will happen within seconds. (**D**) An algorithm that maximizes information gain without considering safety constraints will crash into the comet. (**E**) Our s-FEAST algorithm selects trajectories that have a high likelihood of avoiding the obstacle while also gathering information about the failure. (**F**) The robot is able to successfully identify the underlying failure and return to a trajectory heading away from the obstacle and boundaries. Still frames for all experiment time steps are provided in Appendix C.

S-FEAST represents a substantial improvement over FEAST, by generalizing the fault model to a broader class of partial failures and bias attacks, and extending the theoretical and experimental results. In this chapter, we mathematically formalize the time-critical fault estimation problem subject to state constraints and show that constrained optimization over the coupled fault mode and physical state uncertainty is challenging for existing methods. We address this gap by combining the belief-space tree search and marginalized filtering we developed with FEAST with concentration inequalities to efficiently maximize an information gathering objective and satisfy probabilistic state constraints. Finally, we present theoretical analysis, real-time hardware experiments, and numerical experiments to validate our claims.

3.2 Problem Statement

In this section, we extend the active fault estimation problem to maintain safety while planning actions to diagnose the underlying as quickly as possible. First we generalize general the control-affine system dynamics given by Eqs. 2.14 and 2.15 to include degradation and biases in addition to the binary failures considered in the pervious chapter:

$$x_k = f(x_{k-1}) + B(x_{k-1})\left((\mathbb{I} - \Phi_B)u_k + \Phi_{B,\mathbb{1}}\right) + w_k \tag{3.1}$$

$$y_k = (\mathbb{I} - \Phi_C)Cx_k + \Phi_{C,\mathbb{1}} + v_k \tag{3.2}$$

$$u_k \in U \subseteq \{0,1\}^m \tag{3.3}$$

The generalized fault model is Φ_B, Φ_C, $\Phi_{B,\mathbb{1}}$, $\Phi_{C,\mathbb{1}} = \mathrm{diag}(\phi_{B/C/B,\mathbb{1}/C,\mathbb{1}})$ representing changes due to degradation or biases in the actuators and sensors:

$$\phi_{B_i} = \begin{cases} 1 & \text{if } i \text{ actuator is completely failed} \\ 0 & \text{if } i \text{ actuator is nominal} \\ a_i & \text{if } i \text{ actuator is partially degraded} \end{cases}, \qquad \phi_B = [\phi_{B_1}, \ldots, \phi_{B_m}] \tag{3.4}$$

$$\phi_{B,\mathbb{1}_i} = \begin{cases} 1 & \text{if } i \text{ actuator is stuck full on} \\ 0 & \text{if } i \text{ actuator is nominal} \\ a_i & \text{if } i \text{ actuator is partially biased} \end{cases}, \qquad \phi_{B,\mathbb{1}} = [\phi_{B,\mathbb{1}_1}, \ldots, \phi_{B,\mathbb{1}_m}] \tag{3.5}$$

where $a_i \in (0,1)$. The sensor fault models $\phi_C, \phi_{C,\mathbb{1}}$ are defined analogously. Both complete failure and partial failure (degradation) cases are considered in this chapter. We again assume the fault state does not change with time, and define the concatenated vector of all faults:

$$\phi_k = \phi_{k-1} = \phi = (\phi_B, \phi_{B,\mathbb{1}}, \phi_C, \phi_{C,\mathbb{1}}) \in \Phi \subset [0,1]^{2(m+p)}, \quad |\Phi| = N_\Phi < \infty \tag{3.6}$$

where Φ is the set of N_Φ considered faults that live in the continuous space of $2(m+p)$ dimensional vectors with elements restricted between 0 and 1. Note that this definition of Φ is a super set of the definition in the pervious chapter. For this reason, we will not introduce a new symbol, but will use the general fault model for the rest of this chapter when referring to ϕ, Φ, or the augmented state q.

We use a standard superlevel set notion of probabilistic safety:

Definition 7 (α-Safety)**.** *Consider a set of safety constraints on the physical state, $\{g_i\}$ that must all be simultaneously satisfied for a system to be safe ($g_i(x) \geq 0, \forall i$). Define the safety function h as $h(x) = \min_i g_i(x)$ and the corresponding set of safe physical states X_h as $X_h = \{x \mid h(x) \geq 0\}$. Define the set of α-safe beliefs $\mathcal{B}_{h,\alpha} \subseteq \mathcal{B}$, as the beliefs in which the physical state has a probability of at least α of being safe with respect to h:*

$$\mathcal{B}_{h,\alpha} = \{b \in \mathcal{B} \mid \int_X b(x) \mathbb{1}_{X_h}(x) \mathrm{d}x \geq \alpha\} \tag{3.7}$$

where the indicator over the set of safe states $\mathbb{1}_{X_h}(x) = 1$ if $x \in X_h$ and 0 otherwise. Similarly, the indicator over α-safe beliefs $\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) = 1$ if $b_k \in \mathcal{B}_{h,\alpha}$ and 0 otherwise.

We can now define the safe active fault estimation problem using the same value function as in the active fault estimation, but now subject to an additional probabilistic constraint:

Definition 8 (Safe active fault estimation)**.** *The safe active fault estimation problem for a given safety function, h and safety threshold α, is a partially-observable optimal control problem subject to the constraint that each belief is α-safe.*

$$\pi^*(b_0) = \underset{\pi \in \Pi}{\arg\max} \; V^\pi(b_0) \quad \text{s.t.} \quad \mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi, b_0] = 1 \; \forall k \tag{3.8}$$

where the expectation is across the stochastic policy, measurement and process noise sequences. The corresponding optimal value is $V^*(b_0)$.

Throughout the rest of this and the next chapter, V and V^* refer to this constrained optimization problem. As we did for the active fault estimation problem in Chapter 2, we would like to reformulate Definition 8 into a POMDP which is compatible with our MCTS based algorithm. However, the constraint imposed by the condition that $\mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi, b_0] = 1 \; \forall k$, represents a challenge. The POMDP and BMDP forms we consider in Definitions 4 and 5 are unconstrained, as are the POMDPs considered by FEAST and the tree search algorithm it adapts to information gathering problems [37]. Addressing this challenge is one of our core theoretical contributions, which we present in the next section.

3.3 Methods

In this section we develop s-FEAST as solution to the safe active fault estimation problem (Definition 8). First we provide a high-level overview of our approach, before developing the theoretical guarantees of our safety method and the detailed augmentations we make to FEAST to produce the s-FEAST algorithm.

Safety Condition

Theoretical analysis

First, we reformulate the constrained problem into an equivalent unconstrained problem. This step is necessary because standard Monte Carlo tree search techniques do not explicitly handle constraints [35]. This argument is similar to that presented in convex optimization [45] with log-barrier objective reformulations, except we use an affine objective reformulation that produced empirically higher-performing results for tree search.

The transformed reward function and corresponding value function is defined as follows:

$$R_{h,\alpha}(b_k) = \mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k)\,(r_0 + (1 - r_0)R(b_k)) \tag{3.9}$$

$$V^{\pi}_{h,\alpha}(b_k) = \mathbb{E}\left[\sum_{k=1}^{K} R_{h,\alpha}(b_k) \mid \pi, b_0\right] \quad \text{s.t. Eqs. (3.1), (3.2), (2.6)}, \quad u_k \sim \pi(b_{k-1})\ \forall k \tag{3.10}$$

where $r_0 = \frac{K}{K+1}$ and the expectation is over the noise processes and stochastic policy.

Our first result is that the solution of the transformed problem is equivalent to the solution of the original problem (Definition 8), when one exists. This is formalized with the following theorem:

Theorem 2 (Equivalent unbounded reformulation)**.** *If a global optimal solution, $\pi^*(b_0)$, exists to the constrained safe active fault estimation problem, Definition 8, then the solution of the following unconstrained problem with the transformed value function given by Eq.* (3.10)*, is also a global optimal solution of Definition 8:*

$$\pi^*_{h,\alpha}(b_0) = \arg\max_{\pi\in\Pi} V^{\pi}_{h,\alpha}(b_0) \tag{3.11}$$

Proof. The equivalence of the problems is shown if the optimal policy of the reformulated problem given by Eq. (3.11) has the same value on the original problem

(Definition 8) as the optimal policy given by Eq. (3.8). The reformulated problem is constructed such that any policy resulting in an expected α-safe trajectory (Definition 7) has a minimum expected cumulative reward of Kr_0, which is higher than the maximum expected cumulative reward of an trajectory expected to be unsafe, $K-1$; as $Kr_0 = \frac{K^2}{K+1} > \frac{K^2-1}{K+1} = K-1$.

$$\mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi, b_0] = 1 \; \forall k \iff V^{\pi}_{h,\alpha}(b_0) \geq Kr_0 \tag{3.12}$$

$$\exists k : \; \mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi, b_0] \neq 1 \iff V^{\pi}_{h,\alpha}(b_0) < Kr_0 \tag{3.13}$$

By assumption, the policy $\pi^*(b_0)$, a global optimal solution to Definition 8, exists and is feasible. From the constraints of Eq. (3.8), this solution must satisfy:

$$\mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi^*, b_0] = 1 \; \forall k \implies V^{\pi^*}_{h,\alpha}(b_0) \geq Kr_0 \tag{3.14}$$

Since $\pi^*_{h,\alpha}(b_0)$ is the optimal solution to the reformulation given by Eq. (3.11):

$$V^{\pi^*_{h,\alpha}}_{h,\alpha}(b_0) \geq V^{\pi}_{h,\alpha}(b_0), \forall \pi; \;\; V^{\pi^*_{h,\alpha}}_{h,\alpha}(b_0) \geq Kr_0 \implies \mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi^*_{h,\alpha}, b_0] = 1 \forall k \tag{3.15}$$

So $\pi^*_{h,\alpha}(b_0)$ satisfies the original problem constraints (Definition 8). Because $\pi^*_{h,\alpha}(b_0)$ is admissible for the original problem, by optimality of π^*, $V^*(b_0) \geq V^{\pi^*_{h,\alpha}}(b_0)$. Note that for trajectories satisfying the original problem constraints (Definition 8), the transformed objective is an affine transformation and is monotonic, so with Eq. (3.15):

$$V^*(b_0) \geq V^{\pi^*_{h,\alpha}}(b_0) \implies V^{\pi^*}_{h,\alpha}(b_0) \geq V^{\pi^*_{h,\alpha}}_{h,\alpha}(b_0) \implies V^{\pi^*}_{h,\alpha}(b_0) = V^{\pi^*_{h,\alpha}}_{h,\alpha}(b_0) \tag{3.16}$$

Further, the argument of the extrema is preserved, therefore, $\pi^*_{h,\alpha} = \pi^*$ when there is a unique optimal solution.

□

This reformulation makes the safe active fault estimation probelm compatible with MCTS methods. However, to ensure safety, we need to evaluate the indicator function $\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k)$ throughout the tree search. For a general probability distribution, this function is difficult to compute exactly. Instead, for computational efficiency, we will develop our conservative sampling based approximation. Our approach is based on the following finite sample approximation Chebyshev's Inequality, first developed by Saw *et al.* [46] and simplified by Kabán [47]:

$$\mathbb{P}\left(|Z - \hat{\mu}_Z| > \lambda \hat{\sigma}_Z\right) \leq \frac{1}{M+1} \left\lfloor \frac{M+1}{M} \left(\frac{(M-1)}{\lambda^2} + 1 \right) \right\rfloor \tag{3.17}$$

where Z is a random variable, and λ is a user-specified scalar. The bound is computed by taking M samples that are weakly exchangeable (i.i.d. is sufficient but not necessary) with the random variable to compute the empirical average and standard deviation $\hat{\mu}_Z, \hat{\sigma}_Z$. This bound holds for unknown distributions when $M \geq 2$ and $\lambda \geq 1$. For general random variables, the Chebyshev inequality can be shown to be a tight bound [47], making it well suited to general distributions.

In our setting, the random variable of interest is the safety function applied to a sample from the physical state belief: $h(x)$ where $x \sim b(x)$. To compute the empirical average ($\hat{\mu}_h$) and standard deviation ($\hat{\sigma}_h$) of this safety value, let $x_1, \ldots, x_M$ be i.i.d. samples of $b(x)$. We then have:

$$\hat{\mu}_h = \frac{1}{M}\sum_i h(x_i), \quad \hat{\sigma}_h^2 = \frac{M+1}{M(M-1)}\sum_i (h(x_i) - \hat{\mu}_h)^2 \tag{3.18}$$

Our safety condition then follows directly from applying the finite sample Chebyshev inequality given by Eq. (3.17) to bound the tail of h that is less than zero (the unsafe tail).

Theorem 3 (Conservative sampling bound)**.** *For $M > 2$, a belief $b(x)$, safety function h, $\hat{\mu}_h$ and $\hat{\sigma}_h$ defined according to Eq.* (3.18)*, and $\hat{\mu}_h \geq \hat{\sigma}_h$; satisfying the approximate safety condition of Eq.* (3.19) *indicates that the belief is conservatively α-safe.*

$$\frac{1}{M+1}\left\lfloor \frac{M+1}{M}\left(\frac{\hat{\sigma}_h^2(M-1)}{\hat{\mu}_h^2} + 1\right)\right\rfloor \leq 1 - \alpha \implies b \in \mathcal{B}_{h,\alpha} \tag{3.19}$$

Proof. Defining the physical state associated with the belief as: $Z \sim b(x)$, a random variable, the condition for α-safety is $\mathbb{P}\left(h(x) \geq 0\right) \geq \alpha \implies \mathbb{P}\left(h(x) < 0\right) \leq 1-\alpha$. Adding and subtracting the empirical mean and upper bounding the one sided tail probability with a two-sided condition we have:

$$\mathbb{P}\left(h(x) < 0\right) = \mathbb{P}\left(h(x) - \hat{\mu}_h < -\hat{\mu}_h\right) \leq \mathbb{P}\left(|h(x) - \hat{\mu}_h| > \hat{\mu}_h\right) \tag{3.20}$$

Using Eq. (3.17), and choosing $\lambda = \hat{\mu}_h/\hat{\sigma}_h$ we have:

$$\mathbb{P}\left(|h(x) - \hat{\mu}_h| > \hat{\mu}_h\right) \leq \frac{1}{M+1}\left\lfloor \frac{M+1}{M}\left(\frac{\hat{\sigma}_h^2(M-1)}{\hat{\mu}_h^2} + 1\right)\right\rfloor \tag{3.21}$$

Combining Eqs. (3.20) and (3.21), we arrive at the desired result. □

In general, the condition presented in Theorem 3 is conservative; it is possible for a solution to be α-safe and violate the approximate safety condition (Eq. (3.19)). The slackness comes from two sources, (i) the finite-sample approximation of the Chebyshev inequality and (ii) the potential slackness of the Chebyshev bound itself in the infinite-sample limit. In our experiments, we found that we can effectively eliminate the first source of slackness with $M = 100$ samples. For this reason, we focus on the second source and the effect of this slackness on the optimal solution.

In the infinite-sample limit, $\hat{\mu}_h, \hat{\sigma}_h$ converge to the true statistics μ_h, σ_h and Eq. (3.17) becomes the Chebyshev inequality. We formalize the slackness in the Chebyshev bound with the following lemma, which states that the set of beliefs that satisfy the Chebyshev bound are a well-defined subset of the α-safe beliefs:

Lemma 2 (Conservative α-safe set)**.** *For any belief b, and safety function h with corresponding statistics μ_h, σ_h; there exists a conservatively α-safe set $\tilde{\mathcal{B}}_{h,\alpha} \subseteq \mathcal{B}_{h,\alpha}$, such that the following safety condition is necessary and sufficient for membership:*

$$\frac{\sigma_h^2}{\mu_h^2} \leq 1 - \alpha \iff b \in \tilde{\mathcal{B}}_{h,\alpha} \tag{3.22}$$

Proof. We note for a given belief, the mean and standard deviation are deterministic, so $\tilde{\mathcal{B}}_{h,\alpha} = \{b \in \mathcal{B} : \frac{\sigma_h^2(b)}{\mu_h^2(b)} \leq 1 - \alpha\}$ is well defined. In the limit as $M \to \infty$, Eq. (3.19) becomes $\frac{\sigma_h^2}{\mu_h^2} \leq 1 - \alpha$ so by Theorem 3 or Chebyshev's inequality we have $\forall b \in \tilde{\mathcal{B}}_{h,\alpha}, b \in \mathcal{B}_{h,\alpha}$, so $\tilde{\mathcal{B}}_{h,\alpha} \subseteq \mathcal{B}_{h,\alpha}$. □

To account for the slackness in our safety condition, we modify our reward and value functions as:

$$\tilde{R}_{h,\alpha}(b_k) = \mathbb{1}_{\tilde{\mathcal{B}}_{h,\alpha}}(b_k)\,(r_0 + (1 - r_0)R(b_k)) \tag{3.23}$$

$$\tilde{V}^{\pi}_{h,\alpha}(b_k) = \mathbb{E}\left[\sum_{k=1}^{K} \tilde{R}_{h,\alpha}(b_k) \mid \pi, b_0\right] \text{ s.t. Eqs. (3.1), (3.2), (2.6)} \tag{3.24}$$

Formalizing the slackness of our conservative sampling bound allows us to present a further problem reformulation:

Definition 9 (Conservative Safe Active Fault Estimation)**.** *The conservative safe active fault estimation problem is defined as follows:*

$$\tilde{\pi}^*_{h,\alpha}(b_0) = \arg\max_{\pi \in \Pi} \tilde{V}^{\pi}_{h,\alpha}(b_0) \tag{3.25}$$

with corresponding optimal value $\tilde{V}^*_{h,\alpha}(b_0)$.

The desired behavior of this reformulation is that if the solution of the original problem lies in the feasible space of the conservative problem reformulation, solving the conservative problem will produce the original solution. This property is formalized in the following theorem:

Theorem 4 (Problem reformulation equivalence)**.** *If an admissible policy, $\pi(b_0)$, to the safe active fault estimation problem (Definition 8) exists and satisfies:*

$$\mathbb{E}[\mathbb{1}_{\tilde{\mathcal{B}}_{h,\alpha}}(b_k) \mid \pi, b_0] = 1 \; \forall k \tag{3.26}$$

where $\tilde{\mathcal{B}}_{h,\alpha}$ is given by Lemma 2, then an optimal policy, $\tilde{\pi}^*_{h,\alpha}(b_0)$, to the conservative safe active fault estimation problem (Definition 9) is a sub-optimal solution of Definition 8 constrained to $\tilde{\mathcal{B}}_{h,\alpha}$. Further, if an optimal policy, $\pi^*(b_0)$, to Definition 8 exists and satisfies Eq. (3.26)*, $\tilde{\pi}^*_{h,\alpha}(b_0)$ is an optimal solution to Definition 8.*

Proof. We start with the second claim. By assumption, $\pi^*(b_0)$ exists, is optimal over $\mathcal{B}_{h,\alpha}$, and satisfies $\mathbb{E}[\mathbb{1}_{\tilde{\mathcal{B}}_{h,\alpha}}(b_k) \mid \pi^*, b_0] = 1 \; \forall k$. From Theorem 2, this means that $\pi^*(b_0)$ is optimal on the the conservative safe active fault estimation problem (Definition 9). From the optimality of $\tilde{\pi}^*_{h,\alpha}(b_0)$ on this problem, $\tilde{V}^{\pi^*}_{h,\alpha}(b_0)$ $= \tilde{V}^*_{h,\alpha}(b_0)$ and $\mathbb{E}[\mathbb{1}_{\tilde{\mathcal{B}}_{h,\alpha}}(b_k) \mid \tilde{\pi}^*_{h,\alpha}, b_0] = 1 \; \forall k$. From Lemma 2, $\tilde{\mathcal{B}}_{h,\alpha} \subseteq \mathcal{B}_{h,\alpha}$, so $\mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \tilde{\pi}^*_{h,\alpha}, b_0] = 1 \; \forall k$, so $\tilde{\pi}^*_{h,\alpha}$ is an admissible solution to the safe active fault estimation problem (Definition 8) and is optimal, as it has the same value as an optimal solution.

For the first claim, when a feasible policy of Definition 8 generates expected beliefs in $\tilde{\mathcal{B}}_{h,\alpha}$ ($\exists \pi$ satisfying Eq. (3.26)), restricting the safe active fault estimation problem to $\tilde{\mathcal{B}}_{h,\alpha}$ provides a new safe active fault estimation problem with at least one feasible solution. Optimality of $\tilde{\pi}^*_{h,\alpha}(b_0)$ for this problem then follows from Theorem 2. □

We can finally reformulate our safe and conservative safe active fault estimation problems in to POMDP forms compatible with our MCTS methods, adopting the approach of Lemma 1 in the previous chapter.

Lemma 3 (Equivalent POMDP Reformulation)**.** *If a global optimal solution exists to the safe active fault estimation problem, Definition 8, then the solution of the following POMDP is also a global optimal solution of Definition 8 in the discretization limit $\hat{Y} \longrightarrow Y$:*

$$\begin{aligned}
&\langle Q, U, \hat{Y}, R_{h,\alpha}, T, Z \rangle, \\
&T(q_k, u_k, q_{k-1}) = \mathcal{N}\left(f(x_{k-1}) + B(x_{k-1})\left((\mathbb{I} - \Phi_B)u_k + \Phi_{B,\mathbb{1}}\right), \Sigma_w\right)(x_k) \\
&Z(q_k, u_k, y_k) \propto \mathcal{N}\left((\mathbb{I} - \Phi_C)Cx_k + \Phi_{C,\mathbb{1}}, \Sigma_v\right)(y_k)
\end{aligned} \tag{3.27}$$

where Q, U, $\hat{Y}$, Z, and the belief updates are the same as in Lemma 1, and $R_{h,\alpha}$ is given by Eq. (3.9)*. Further, when $R_{h,\alpha}$ is replaced by $\tilde{R}_{h,\alpha}(b_k)$ given by Eq.* (3.25)*, then the global optimal solution of this POMDP is a global optimal solution of the conservative safe active fault estimation problem (Definition 9).*

Proof. By assumption the policy $\pi^*(b_0)$, a global optimal solution to the original problem (Definition 8), exists and is feasible. The equivalence of the two problems is shown if $\pi^*(b_0)$ is also an optimal solution of this POMDP. By Theorem 2, the existence of $\pi^*(b_0)$ indicates the existence of $\pi^*_{h,\alpha}(b_0)$, an optimal solution to Definition 8 satisfying:

$$\pi^*_{h,\alpha}(b_0) = \underset{\pi \in \Pi}{\arg\max}\, V^{\pi}_{h,\alpha}(b_0) \tag{3.11}$$

So we have the following

$$V^{\pi^*_{h,\alpha}}_{h,\alpha}(b_0) \geq V^{\pi}_{h,\alpha}(b_0) \quad \forall \pi \in \Pi \tag{3.28}$$

Because the POMDP given by Eq. (3.27) also uses $R_{h,\alpha}$ given by Eq. (3.9) as a reward, it shares the value function given by Eq. (3.10) with the reformulation given by Theorem 2 when $\hat{Y} \longrightarrow Y$. Therefore, $\pi^*_{h,\alpha}(b_0)$ and $\pi^*(b_0)$ maximize the value function of Eq. (3.27), so by definition, the solution of Definition 8 is an optimal policy of Eq. (3.27) and the problems are equivalent. The same logic applies when $R_{h,\alpha}$ is replaced by $\tilde{R}_{h,\alpha}(b_k)$, showing equivalence with the conservative safe active fault estimation problem as well. □

Using these reformulations, we develop s-FEAST in the next sections.

Algorithm Overview

An overview of the complete s-FEAST algorithm is shown in Fig. 3.2. As we developed in the previous chapter, our method is an anytime planner based on partially observable Monte Carlo tree search [34], [35], [37] and is diagrammed in Fig. 3.2A. Starting from an initial belief on both the robotic spacecraft's physical and fault states, $b_0(q)$, actions are selected and simulated forward to a planning horizon. The tree explores for actions that both resolve ambiguity in the underlying faults and are predicted to not lead to violations of safety as defined by state constraints. As an anytime algorithm, it refines the simulated futures until interrupted, returning the best action found so far.

In addition to the marginalized filter employed by FEAST [visualized in Fig. 3.2 (B and C)], s-FEAST enforces probabilistic safety constraints with a concentration inequality to provide conservative guarantees of safety for arbitrary belief distributions, noise processes, and safety constraints. Beliefs that satisfy this inequality are assumed to lie within the set of safe beliefs and receive a bonus reward; otherwise they are assigned a reward of zero. As a result, for any safe trajectory the summed reward over the planning horizon is above that of any unsafe trajectory (Fig. 3.2D). With this construction, the convergence of the tree search to the optimal value also ensures safety.

Algorithm Details

We present the pseudocode for s-FEAST in Algorithm 3 below, and discuss the changes with respect to existing belief-space tree-search and FEAST. The differences between s-FEAST and POMCP are again highlighted in blue: (i) when a new node is encountered, the exact Bayesian update is computed with our Marginalized filter, Eq. (2.28); (ii) we use the updated belief to compute exact rewards and generate a value estimate; (iii) we use the exact belief to accurately approximate the safety at each node via our safety condition, Theorem 3.

With the exception of the safety condition and the transformed reward function, s-FEAST is run the same as FEAST. See Chapter 2 for details. Implementation details for each of the experiments is provided in Appendix B.

We can now state the main theorem, which is a direct consequence of reformulating the problem into a search-compatible framework and then applying existing search convergence results: s-FEAST converges to the optimal solutions of the problems given by Definitions 8 and 9.

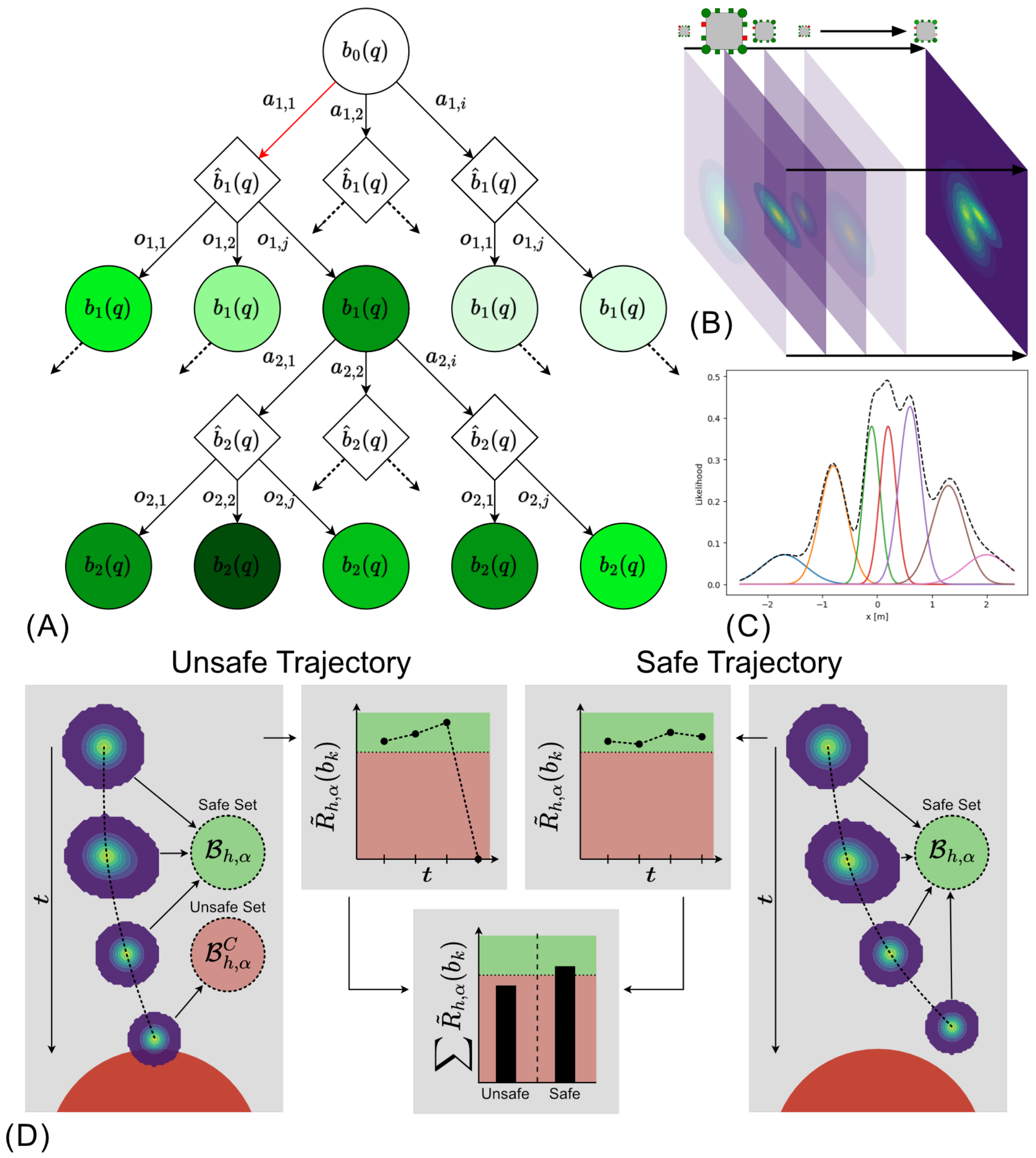


Figure 3.2: **s-FEAST: Method overview.** (**A**) Diagram of the tree search employed by s-FEAST. The tree growth is biased towards nodes leading to better rewards (represented here as darker shading). a represent actions taken, $\hat{b}$ prior beliefs, o observations, and b updated beliefs. (**B**) Illustration of our marginalized filter representing the position of the robotic spacecraft as the sum of estimates conditioned on each possible failure. (**C**) When the physical estimators are Kalman Filters, the marginalized filter of a complicated multi-modal distribution is a combination of Gaussians. (**D**) The belief at each time step can be classified as in or outside of the set of safe beliefs ($\mathcal{B}_{h,\alpha}$) based on bounding the likelihood of collision with the obstacle (shown as a red semi-circle). The reward function used by s-FEAST, $\tilde{R}_{h,\alpha}(b_k)$, results in any trajectory of safe beliefs having a higher cumulative reward than any trajectory with at least one unsafe belief.

Algorithm 3: The POMCP and s-FEAST algorithms for belief-space planning. For this pseudocode, we adapt the original POMCP algorithm to our notation, with modifications made to create s-FEAST highlighted in blue [37]. MF refers to our marginalized filter, apxSafety refers to the approximate safety condition given by Eq. (3.19) and Theorem 3.

```
globals: $\hat{V}(\cdot) \leftarrow 0,\ N(\cdot) \leftarrow 0$
def search($b_0$):
    for $i \leftarrow 1$ to $N$ do
        simulate($q \sim b_0, \emptyset, 0, b_0$);
    return $\arg\max_u \hat{V}(H \cup \{u\})$;
def safe($b$):
    for $i \leftarrow 1$ to $M$ do
        $x_i \sim b$;
        $h_i \leftarrow h(x_i)$;
    $\hat{\mu}_h, \hat{\sigma}_h \leftarrow$
      sampleStatistics($\{h_1, ..., h_M\}$);
    return apxSafety($\hat{\mu}_h, \hat{\sigma}_h, M, \alpha$) ;
def simulate($q_d, H_d, d, b(H_d)$):
    if $d > K$ then
        return 0;
    $u_{d+1} \leftarrow \arg\max_u$
      $\hat{V}(H_d \cup u) + c\sqrt{\frac{\log N(H_d)}{N(H_d \cup u)}}$;
    $(q_{d+1}, y_{d+1}) \sim G(q_d, u_{d+1})$;
    $H_{d+1} \leftarrow H_d \cup \{u_{d+1}, y_{d+1}\}$;
    if s-FEAST then
        $b(H_{d+1}) \leftarrow$
          MF($b(H_d), u_{d+1}, y_{d+1}$) ;
    else
        $b(H_{d+1}) \leftarrow b(H_{d+1}) \cup q_{d+1}$;
    $r \leftarrow R(b(H_{d+1}))$;
    if s-FEAST then
        $r \leftarrow$ safe($b(H_{d+1}), h, \alpha$) $*$
          $(r_0 + (1 - r_0)r)$ ;
    $r \leftarrow r + \gamma$ simulate($q_{d+1}, H_{d+1}$,
      $d + 1, b(H_{d+1})$);
    if $N(H \cup u_{d+1}) = 0$ and $N(H_{d-1} \cup$
      $u_d) = 0$ and not s-FEAST then
        return $r$
    $N(H_d) \leftarrow N(H_d) + 1$ ;
    $N(H_d \cup u_{d+1}) \leftarrow N(H_d \cup u_{d+1}) + 1$ ;
    $\hat{V}(H_d \cup u_{d+1}) \leftarrow$
      $\hat{V}(H_d \cup u) + \frac{r - \hat{V}(H_d \cup u_{d+1})}{N(H_d \cup u_{d+1})}$ ;
    return $r$;
```

Theorem 5 (Optimality of s-FEAST)**.** *Let $\mu(b_0)$ denote the policy produced by s-FEAST, and $\tilde{\pi}^*_{h,\alpha}(b_0)$ denote an optimal policy to the conservative safe active fault estimation problem (Definition 9). In the limit of $M \to \infty$, the value of these policies converge:*

$$\lim_{N\to\infty} \left(\tilde{V}^{\mu}_{h,\alpha}(b_0) - \tilde{V}^{*}_{h,\alpha}(b_0)\right) \to 0 \tag{3.29}$$

with convergence rate $O(\log N/N)$. Further, if an optimal policy, $\pi^*(b_0)$, to Definition 8 exists and satisfies Eq. (3.26)*, $V^{\mu}(b_0)$ converges to $V^*(b_0)$.*

Proof. From Lemma 2 we have that in the limit of $M \to \infty$, $\tilde{\mathcal{B}}_{h,\alpha}$ is the set for which Eq. (3.22) is a necessary and sufficient condition for membership, and $\tilde{\mathcal{B}}_{h,\alpha} \subseteq \mathcal{B}_{h,\alpha}$. From Theorem 4 we have that if s-FEAST solves the conservative safe active sensing problem (Definition 9) with the stated convergence rate, we achieved both the claimed results.

To show s-FEAST solves Definition 9, we note that Definition 9 can be equivalently reformulated as a POMDP by Lemma 3. To solve the equivalent POMDP, s-FEAST employs the marginalized filter given by Eq. (2.28) to perform an Bayesian update when creating a new node in the tree search, and incurs an accurate reward. Therefore, we are performing PO-UCT from (69) and inherit the convergence rate from their Theorem 1. □

3.4 Robotic Spacecraft Simulator Hardware Experiments

We implemented s-FEAST on the M-STAR robot [43], [44] in the same manner as our FEAST experiments described in Section 2.6.

The robot was tasked to diagnose sensing and actuation faults while on a collision course with our model comet (Fig. 3.1C). The true failure was the loss of both retro thrusters (Fig. 3.1B), which required the M-STAR robot to reorient before it was able slow down and stabilize itself. The safety constraints were to avoid the comet obstacle as well as the walls of the simulator room, shown as the red regions in Fig. 3.1 (D to F), with a 90% or higher probability. With these settings, s-FEAST was able to successfully identify the true failure state while maintaining safety, validating our approach on hardware. These experiments demonstrated that considering safety or fault estimation alone cannot solve this problem (Fig. 3.1D), while s-FEAST can reliably plan evasive actions under uncertain component failure [Fig. 3.1 (E and F)]. A complete time series of s-FEAST and the baseline methods is presented in Appendix C.

For these experiments, a computational budget of 0.78 seconds on a 1.10 GHz, 4 core CPU (i5-1035G4) was used, which typically resulted in 85 simulations per time step. We show in numerical experiments this is sufficient computation to substantially improve over existing approaches. More details of our real-time implementation and performance are provided in Appendix C.

3.5 Numerical Simulations

To validate our algorithm quantitatively, we considered s-FEAST in four safety-critical scenarios against baselines of Sequential Convex Programming (SCP), Discrete Control Barrier Functions (D-CBF), greedy, and random policies. Each simulation was performed on a three degree of freedom model of the M-STAR robot. We evaluated each algorithm over 1000 trials according to the fraction of trials safe throughout the experiment as well as the product of a diagnostic reward and success rate we used in our FEAST evaluations. In these experiments however, the simulation does not terminate early when a diagnosis is returned or when the safety conditions are violated, to demonstrate the ability to maintain safety over the full experiment. A timestep during an experiment trajectory is only considered safe if every previous timestep was as well. Because of this, the average safety of a policy is monotonically decreasing

In these simulations, numerical instability of the Extended Kalman Filters (EKF) used to update the belief between states occasionally leads to invalid belief-states (NaN). If the the previous fault estimate is at least 95% confident, this diagnosis is accepted for the remainder of the experiment. Otherwise the experiment is considered to have failed to diagnose the fault and receives no further reward. When computing the standard deviation of the rewards, these experiments are removed to avoid biasing, but the trial is considered unsuccessful for the diagnostic success rate computation.

Details on the systems considered, as well as additional numerical results are provided in Appendices B and C, respectively.

Overview of Baselines

We provide a brief overview of the baselines (random, greedy, D-CBF, SCP) we compared against in the following simulation results. The selection of baselines was designed such that s-FEAST and these baselines covered a permutation of deterministic vs. probabilistic state representations and greedy vs. planning algorithmic implementations, with s-FEAST as the probabilistic planning solution. All methods used the same estimator between time steps and each baseline solved for the next action to take. Implementation details are provided in Appendix B.

The random and greedy baselines are the same as seen in our validation of s-FEAST, with the modification that the greedy, active approach considers the best safe action to take. As before, it only considered a lookahead horizon of one and did not resample any actions, which made it vulnerable to near term danger and outlier simulations. The random baseline did not consider safety at all. Together, the random and greedy baselines served to illustrate the shortcomings of random and one-step planning approaches in identifying the underlying faults when safety constraints must also be satisfied.

The next two baselines were deterministic safe control methods. The discrete control barrier function (D-CBF) [48] method acts greedily, considering only the safety of the next time step, whereas the sequential convex programming (SCP) [49], [50] method plans safe trajectories over a horizon. These algorithms do not have a probabilistic representation of the state or system model and require a fully-observable state. Work has been done to extend both methods to consider stochastic noise via chance constraints for SCP [51] and probabilistic safety bounds for D-CBF [52], though neither method is compatible with the coupled fault mode and physical state uncertainty considered here. To adapt them to our partially observable setting, we used the most likely failure state and corresponding mean position estimate as the assumed system dynamics and initial position and added a buffer to each obstacle. It should be noted that when the system model is accurately known, a controller satisfying the D-CBF condition renders all states in the safe set forward invariant and therefore safe. However, this is not guaranteed if the most likely model is inaccurate, and we saw this method fail in our simulations for this reason. These control baselines served to illustrate the limitations of the control-estimation separation principle in safety-critical fault estimation problems.

Overview of Scenarios

In each of the following scenarios, we considered a robotic spacecraft initially 10 m from a circular obstacle which represented some target of interest the robot was investigating before the failure occurred. For s-FEAST and our safety-aware baselines, we imposed a chance constraint that with 90% or higher probability, the spacecraft must avoid collision and deviate no more than 25 m in any direction from its initial position at each time step. In practice, we saw this chance constraint enabled s-FEAST to achieve 90% or higher safety throughout the experiment, as the robot was near the obstacles or bounds for only a few time steps.

To highlight various sources of difficulty our method addresses, we considered two fault cases in two increasingly difficult initial conditions. Binary faults, where components either worked or were completely failed, illustrated the challenges posed when components fail silently, resulting in ambiguity between fault models. Alternatively, continuous component degradation and biases presented a larger challenge for safety, as actuator biases could destabilize a system if unaddressed.

Scenario: Binary Fault Diagnosis in Proximity to an Obstacle

In the first scenario, the spacecraft started with no initial velocity, and up to three components completely failed, where the underlying binary failure was randomly selected for each trial. This case was selected to examine how well each policy achieves our desired 90% chance of safety when the spacecraft was not in any immediate danger, and demonstrate how naive information gathering could put the system at risk. The results are summarized in Fig. 3.3A.

Examining the safety of each method, we see that the greedy and random baselines dramatically underperformed the other methods. This was observed to be in part due to their inability to consider safety beyond the next time step or, in the case of the random baseline, at all. This led to destabilizing actions being selected more often, making future time steps more likely to have no safe action available.

Considering the reward and diagnostic success of each method, the s-FEAST algorithms all outperformed the CBF and SCP baselines, as did the random and greedy baselines. This was due to the CBF and SCP baselines not taking any information gathering actions, or any actions at all until the system was close to becoming unsafe. So both baselines typically failed to diagnose the underlying failure by the end of the experiment, which lead to low diagnosis success rates of 20.8% and 19.5%, respectively. The random and greedy algorithms performed similarly to the s-FEAST algorithms in diagnosing the underlying fault, but at the price of considerably worse safety, with final safety values of 17.4% and 16.9% respectively.

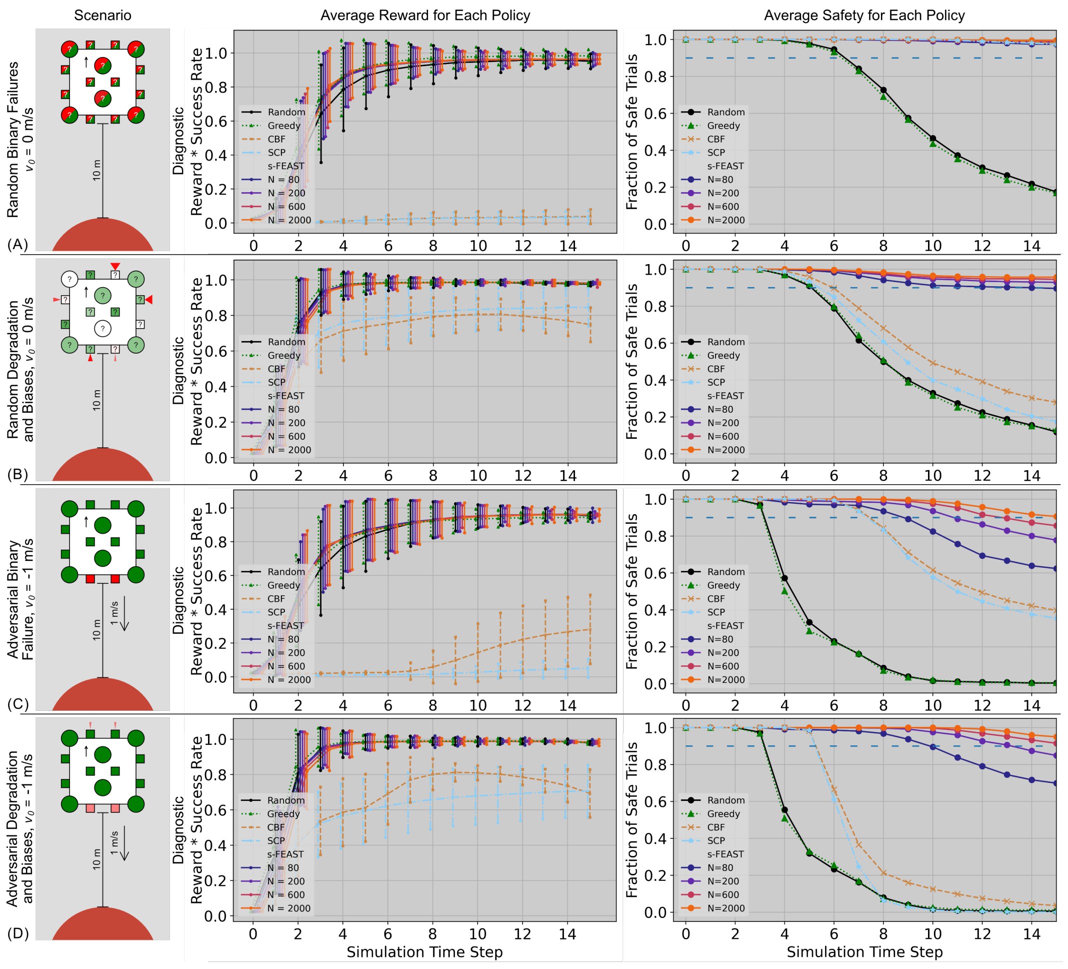


Figure 3.3: **Validation of s-FEAST.** The numerical performance of our algorithm compared with baselines across several scenarios. (**A**) The robotic spacecraft started 10 m from the obstacle with no initial velocity, subject to random binary failures of up to three components. (**B**) Each component was now randomly subjected to continuous degradation or bias, with nominal components more likely. (**C**) The robotic spacecraft now started with an initial 1 m/s velocity towards the obstacle, subject to an adversarial binary failure of its two retro thrusters. (**D**) The adversarial failure was now both retro thrusters degraded by 80%, and both forward thrusters stuck on with a 10% bias. In all experiments, s-FEAST considered 40 possible binary or general faults and started with a uniform prior over all possibilities. In the visualization of the spacecraft component health in the left column, squares represent the thrusters and circles abstractly represent the position and orientation sensors. Green components are healthy, red are failed, and red actuations represent bias thrust of varying degrees (sensor bias is not visualized). Note that for readability, the data for each time step is artificially spread out horizontally.

Scenario: Continuous Degradation and Bias Fault Diagnosis in Proximity to an Obstacle

In this experiment, we considered the same scenario as before, but now components could be partially degraded, giving only a fraction of their nominal output. This could correspond to actuator damage resulting in decreased efficiency or a miss-calibrated sensor. Components could also be subject to constant biases, correlating to unexpected behavior such as an actuator stuck on, sensor offset, or even malicious signal injection. As before, we assumed the fault was constant for the duration of our diagnosis period. Faults were generated by sampling eight unique biases with five component degradations each, for a total of 40 possible faults as before. The true fault was set to one of these. Additional implementation details are provided in Appendix B.

The results of this simulation are shown in Fig. 3.3B. Compared to the previous scenario, we see similar relative behavior, and all methods had a higher diagnostic reward and a lower safety. The diagnostic reward increased because the faults were no longer silent. Any bias injected a signal into the system enabling passive identification of these faults. However, the active signal made enforcing safety more challenging, as bias acceleration could lead to constraint violations. This trade off was seen through a drop in safety for all policies. For example, from time step 3 to 4, the deterministic methods (SCP and CBF) started to decline in safety, whereas the diagnostic reward increased more rapidly than the s-FEAST methods for the first time. This trend continued through the experiment, with reward increasing but safety dropping.

The ambiguity in component degradation for a given bias provides a likely explanation for this trend. Since neither SCP or CBF methods consider a belief, information gathering to resolve this ambiguity could not be explicitly performed. This could result in an incorrect assessment of both the safety of the current state as well as the control authority if actuator faults were not yet detected or resolved. When actions were taken to avoid collision, they may have occurred too late or with unexpectedly small effect, leading to safety violation, but also yielding more information on the component degradation, giving an increase in diagnosis reward. Finally, we note the decrease in diagnostic reward for the CBF method near the end of the experiment stems from filter divergence. This was due to large control inputs leading to numerical instability in the Extended Kalman Filter without converging to a fault estimate.

Scenario: Collision Course Under Adversarial Binary and Continuous Failures

In the final two scenarios examined, the spacecraft was now subjected to the same underlying fault in every trial and was initialized on a collision course with an obstacle. We considered an adversarial failure for both our binary and continuous degradation and bias scenarios. In the binary scenario, the two retro thrusters on the spacecraft were completely off, and in the continuous case, the retro thrusters were subject to an 80% degradation and the forward thrusters were subject to a 10% bias. In both cases, the spacecraft had to first change orientation, then slow down to reliably avoid a collision. In addition, the spacecraft still started with a uniform prior over 40 possible failures, so it had to take actions to reduce the risk of collision before fully identifying the underlying fault. Since this behavior required planning over a horizon, we considered these to be adversarial faults for this scenario and chose this scenario to demonstrate s-FEAST's robustness to outlier failures that posed an outsized risk to the system.

The results are shown in Fig. 3.3 (C and D), where we see that all baselines now achieved less than 40% safety in the binary case (CBF: 39.7% , SCP: 35.4%, Random: 0.4%, Greedy: 0.5%) and less than 4% in the continuous case (CBF: 3.6% , SCP: 0.1%, Random: 0.5%, Greedy: 1.1%), and were outperformed by s-FEAST with even the lowest level of planning. These results suggest that running as little as $N = 80$ simulations can achieve a final safety rate of 62.4% in the binary case and 69.9% in the continuous case. For $N = 200$, the safety rate was 77.8% and 84.9% for the binary and continuous faults. We used this result to inform our real-time hardware experiments, where the typical planning amount was $N = 85$ simulations per tree due to a tight computational budget. The reward for the hardware algorithm shown in Fig. 3.1 was similar to that predicted by this simulation experiment.

We again see that with binary faults, our CBF and SCP baselines failed to gather any information until collision was imminent, and only gained diagnostic reward as a result of attempting to remain safe. Similarly, in the continuous case, the baseline methods gained some information immediately as a result of the bias signal, but failed to further diagnose until evasive actions were taken, which occurred sooner and at higher speed due to acceleration from the bias input. In the next section, we consider an example to illustrate how s-FEAST succeeded where these baseline methods failed.

Qualitative Interpretation of Tree Data

The tree data structure provides some qualitative interpretability of the inner workings of s-FEAST. In Fig. 3.4, we see the spacecraft initially on a collision course under the adversarial failure of both retro thrusters. This is the same binary crash course scenario examined in the previous subsection, with a higher initial velocity of 2 m/s to better demonstrate the qualitative behavior of our algorithm. Before identifying the underlying failure, s-FEAST selected actions to adjust the spacecraft's trajectory to the side of the obstacle. This turned out to be a necessary strategy in this scenario, as after the failure is identified in the third time step, it took another seven time steps to reorient and come to a stop. This obstacle avoidance behavior was also seen in our hardware experiments, such as in Fig. 3.1.

The baseline methods were unable to discover this behavior, as both the greedy and CBF policies do not consider the possibility of failure beyond the next time step and the SCP policy does not take any information gathering actions so will be unaware of the failure until it attempts and fails to slow down. Like our simulation results, this suggests that proactive information gathering is essential to avoiding model uncertainty in these safety-critical situations, as any unknown component failure can jeopardize the systems performance in unexpected ways.

3.6 Chapter Summary

In this chapter, we generalized the active fault estimation problem from Chapter 2 to additionally consider degradation, actuator and sensor bias, and most importantly, probabilistic safety constraints. We then reformulated the safe active fault estimation problem (Definition 8) to an unconstrained form via Theorem 2. Using Theorem 3 and Lemma 2, we defined a conservative sampling bound and the corresponding $\tilde{\mathcal{B}}_{h,\alpha}$ and formalized the conservative safe active fault estimation problem (Definition 9). Finally, Theorem 4 formalized when the solution to the two problems are equivalent, and Theorem 5 demonstrated convergence of s-FEAST to optimal solutions for each.

We make some remarks on this result: First, despite applying the existing search result from [35] and [37], solving problems with belief-dependent objectives and chance-constraints for general belief distributions represents a new capability enabled by our reformulations. Second, we note that $\tilde{\mathcal{B}}_{h,\alpha}$ is in general unknown, or computationally intractable. However, we do not need to know $\tilde{\mathcal{B}}_{h,\alpha}$, there just needs to exist an admissible solution in $\tilde{\mathcal{B}}_{h,\alpha}$ for s-FEAST to converge. For the safety constraints of interest we investigated, we observed in our simulations that solutions

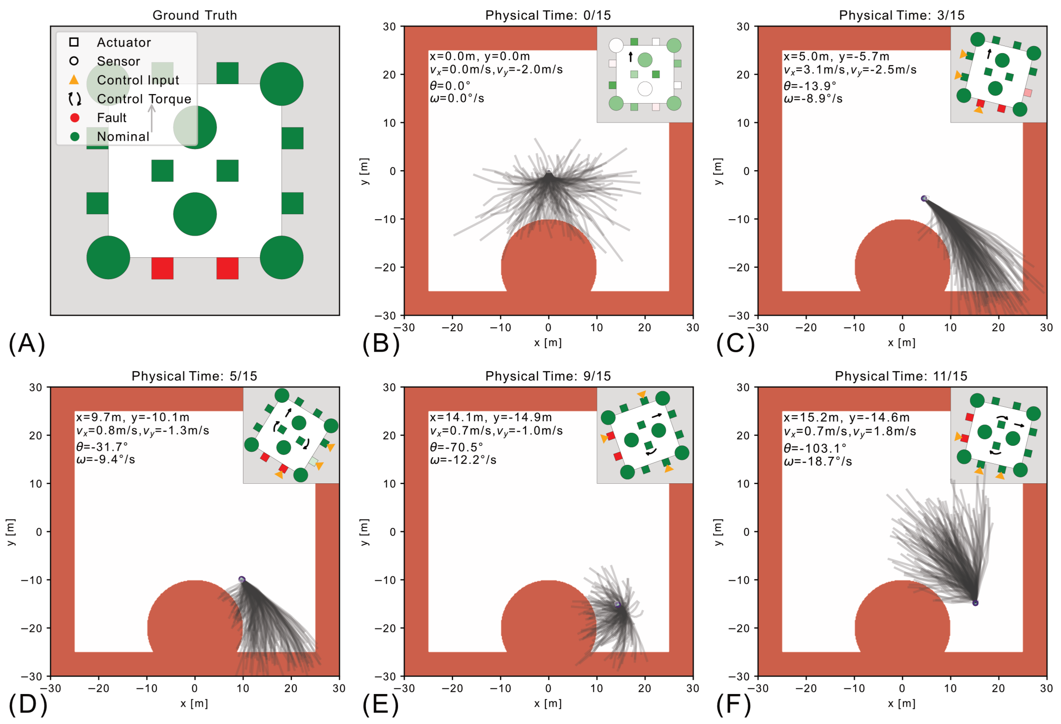


Figure 3.4: **Qualitative analysis of s-FEAST's collision avoidance under an adversarial fault.** (**A**) The two retro thrusters the spacecraft needs to slow down are dysfunctional. (**B**) The spacecraft starts on a collision course with the obstacle and a uniform belief over randomly selected binary failures of actuators (shown as squares) and sensors (abstracted as circles). (**C**) By the third time step, the spacecraft has mostly identified the underlying failure, indicated by the red components. At this point, it has proactively taken action to avoid the obstacle, before the fault was determined. (**D**) After dodging the obstacle, most future trajectories take the spacecraft out of bounds. (**E**) By the ninth time step, the spacecraft has reoriented and started to slow down. (**F**) The spacecraft has reversed course by the eleventh time step and remains safe for the rest of the experiment.

could come close to violating the constraints relative to the size of the safe state space (such as in Fig. 3.4D), indicating that $\tilde{\mathcal{B}}_{h,\alpha}$ is tight. Similarly, we observed empirically that $M = 100$ was sufficient for converged safety estimates. Third, it is possible that the optimal solution to the safe active fault estimation problem lies outside $\tilde{\mathcal{B}}_{h,\alpha}$, and in this case s-FEAST will converge to a sub-optimal approximation of the optimal solution. We argue that the only cases where this occurs is when the optimal trajectory takes the spacecraft close to violating a safety constraint, which while within the bounds of the problem, are the riskiest trajectories.

We believe that s-FEAST balances well the competing interests of safety, performance, and computational complexity. In the next chapter, we will discuss the context of our work with existing methods, as well as the implications for future applications.

Chapter 4

S-FEAST REALTED WORK AND DISCUSSION

[1] J. Ragan*, B. Rivière*, and S.-J. Chung, “Bayesian active sensing for fault estimation with belief space tree search,” *AIAA Scitech 2023 Forum*, 2023. DOI: 10.2514/6.2023-0874,

[2] J. Ragan, B. Rivière, F. Y. Hadaegh, and S.-J. Chung, “Online tree-based planning for active spacecraft fault estimation and collision avoidance,” *Science Robotics*, vol. 9, no. 93, eadn4722, 2024. DOI: 10.1126/scirobotics.adn4722. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.adn4722,

Over the course of the last two chapters, we have developed our FEAST and s-FEAST algorithms and presented the numerical and hardware experiments we used to validate them. In this chapter, we present a detailed comparison of our algorithms with existing methods. We also provide a detailed discussion of the key strength's of our method in comparison with the literature, including how we provide better scaling in information gathering problems. We then discuss our real-time performance, our intended use cases, the applicability of our method to a broader class of problems, and finally the limitations of our method. We will primarily focus our discussion on s-FEAST, as the completed version of our method. However, any discussion about active fault estimation in the absence of safety concerns applies to our FEAST method as well.

4.1 Context with Related Work

Our work sits at the intersection of several fields that can be applied to this problem of safe active fault estimation. We qualitatively summarize them according to three capabilities in Fig. 4.1. First, the flexibility of the system model referring to the linearity of system equations, any assumptions, and the structure of the uncertainty model. Second, the flexibility of the safety condition representing if the method is limited to bounding the expected state alone or if it can also constrain the uncertainty distribution. Third, for methods that select an action to take, we consider their ability to run in real-time. We elaborate on the relevant related work in the following subsections.

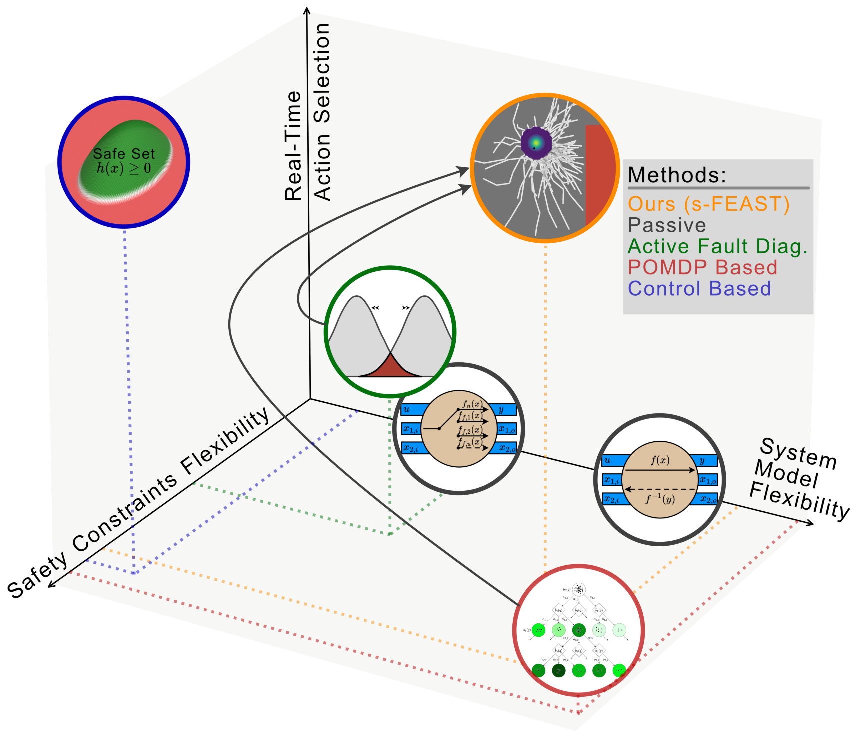


Figure 4.1: **Related work context.** We qualitatively contextualize our work with other relevant approaches applied to fault estimation. Each method is separated by the flexibility of the safety constraints, flexibility of the system model, and the real-time performance for selecting actions. Passive methods such as traditional FDIR [17] do not consider diagnostic actions or safety constraints, and so are restricted to one axis, but represent a broad range of system models. Active fault diagnosis approaches [53] compute inputs to determine between possible underlying faults, but are often limited to specific systems, uncertainty models, or constraints and may lack real-time guarantees. POMDP methods [54], [55] can model a wide range of systems and constraints, but are often computationally intensive to solve, especially in belief-space planning domains. Control based approaches [50], [56] can quickly find actions to satisfy deterministic safety constraints, but traditionally do not consider model uncertainty or information gathering.

Traditional Passive Fault Estimation

Traditionally, system level approaches to fault estimation methods have been passive; actions are not taken to determine the underlying failure but are instead based upon the input-output data during normal operations which is monitored for abnormalities [17]–[19]. This has also been the case for space systems, which have historically used passive FDIR algorithms coupled with safe mode and ground-in-the-loop diagnosis and recovery when faults are detected [57]. Similarly for terrestrial robotics, fault estimation methods were historically passive and designed to alert operators and arrest operations [58], and have been limited by the high levels of ambiguity [59]. In more recent work, passive fault estimation in robotic systems has developed to include data-driven models and account for varying levels of system autonomy [60]–[62] and distributed systems [63]. These passive approaches typically estimate between a small number of possible system fault models with limited noise so are represented as less flexible in Fig. 4.1.

Other approaches to passive fault estimation are constraint-based methods which check the consistency of input and output constraints for each component throughout a system. These have been deployed in several settings due to their simplicity, robustness and ease of user understanding [64], [65]. Although a wide range of systems can be modeled, making them more flexible than other passive methods, these approaches often need to be custom designed for each failure case. They also have limited ability to handle uncertainty and noise, and are potentially not robust to unmodeled scenarios. Since passive FDIR methods do not select actions or consider the safety of those actions, we do not consider them in the context of safety constraint flexibility or real-time planning performance.

Active Fault Diagnosis

One drawback of passive fault estimation is the possibility of multiple plausible fault scenarios. This motivates the problem of selecting control inputs to gather information about the underlying failures. Although optimality conditions can be derived [66], tractable general algorithms do not exist, giving rise to a large body of work in active fault diagnosis [53]. Early work considered discrete-event systems with enumerated states and transitions, designing controllers to ensure the system would be diagnosable [67], with extensions to satellite applications [68], [69].

In linear dynamical systems, the true fault can be determined from multiple possibilities by solving for actions that yield the least expected overlap in hypotheses [70]. Similar to our s-FEAST method, this approach seeks actions that lead to the most useful observations over a horizon. However, it is restricted to systems with Gaussian noise and can only consider constraints on the expected state, whereas s-FEAST can enforce more general chance constraints. Further, globally minimizing this measure of hypothesis overlap or the approximate bounds can become expensive to compute in real-time. An extension of this work [71] has achieved real-time evaluation in relatively low-dimensional problems, but it can only optimize control actions greedily over a single timestep.

Another approach is to design input sequences guaranteed to separate the various fault models in linear systems, provided the disturbances are bounded to zonotopes [72]. One limitation of this approach is the need for the separating inputs to be robust to the worst case disturbances. A closed-loop implementation of this algorithm can lead to less conservative solutions, but may be computationally impractical to run online and require a compromise hybrid offline/online approach to balance conservatism and performance [73]. Similarly, when the uncertainty in model parameters is energy bounded, it is possible to find minimum energy auxiliary signals to distinguish between fault models in linear systems [74], and this approach has been extended to include small, bounded non-linearities [75] and linearizations [76].

These active fault diagnosis approaches demonstrate the usefulness of information gathering applied to fault estimation. However, they are often limited by the types of systems they can be applied to, or make assumptions on the types of uncertainty, restricting model flexibility as represented in Fig. 4.1. Similarly, the constraints considered are often deterministic or only valid in the bounded disturbance case, limiting the flexibility of safety constraints. Finally, many of these methods are computationally intensive and may not provide real-time guarantees. Onboard real-time systems, anytime algorithms that can be interrupted early and return a valid (if sub-optimal) solution are desirable.

Although not directly derived from this body of work, robotic self-modeling, where a robot continuously performs exploratory actions to update its onboard dynamics model [77] is a closely related method. Recently, self-identification has been used to distinguish between multiple possible manipulation models of a robotic hand [78], as well as to learn visual self models [79].

Partially Observable Markov Decision Process Methods

As discussed in Chapter 2, Partially Observable Markov Decision Processes (POMDPs), provide an alternative framework for considering passive and active fault estimation as decision making problems. POMDPs provide a flexible modeling representation, but they are intractable to solve in general, and solutions are often limited to small problems, being performed offline, or inexact methods [54]. For example, a POMDP is used to model the fault estimation problem in [55]. However, the algorithm used to solve the problem only considers the partial observability for the first time step, and cannot perform any active information gathering.

To consider constraints in POMDPS, prior work has added cost terms that must be kept within a specified budget [80]. When this budget is set to zero, these problems can represent hard constraints such as collision avoidance. Offline solutions to this approach include approximate linear programming [81], dynamic programming [82], and gradient ascent with constraint projection [83]. Online methods have also been proposed, including hybrid approaches that consider constraint feasibility up to a sub-horizon and approximate the rest of the planning horizon with an offline estimate [84], an extension of POMCP to discrete constrained POMDPs [85], and a method that extends online solutions to continuous systems by limiting branching [86]. A shared limitation is that these methods constrain only the expected cost, which may not be suitable for risk averse settings or systems with large state estimation uncertainty.

Alternatively, general probabilistic bounds, or chance constraints, can be applied to POMDPS. This approach can be shown to be more general than zero cost constraints [87], and allows for bounds on statistics other than expectations. Approximate offline solutions in this setting include using mixed integer linear programming [88], and pruning high risk branches [87]. To solve chance constrained POMDPs online, a heuristic search that is then iteratively improved in an anytime fashion has been proposed [89], but lacks formal guarantees. Chance constraints have also been applied in belief-space planning for linear Gaussian systems [90] and via log barrier function transformations or soft constraints [91].

Constrained POMDPs are also solvable with model-free approaches trained during an offline phase, such as Dreamer V2 [92] and latent policy optimization [93]. However, compared to online methods, the reliance on an offline training phase makes these methods vulnerable to out-of-domain events [94]. Furthermore, these methods lack theoretical guarantees of optimality convergence and safety assurance, which are especially important for high-cost space missions.

Although tree-based online POMDP solvers such as POMCP [37] work well when the reward is a function of the state, the experiments in Chapter 2 suggest they scale poorly compared to FEAST and s-FEAST in information gathering problems due to the need for particle filter based belief estimates to converge at each node before an accurate value estimate can be made, as we will explore in the next section. This results in the low real-time performance for this setting depicted in Fig. 4.1.

Information Gathering Partially Observable Markov Decision Processes

Information gathering POMDPs use a reward that is directly dependent on the belief, whereas standard POMDPs use a probability-weighted average of state-dependent rewards. Previous work in information gathering POMDPs typically does not consider estimating the system dynamics (such as the fault state) or safety constraints, so we consider them separately from the related work we contextualize in Fig. 4.1. POMDP solvers can be guided toward information gathering behavior via sub-goal states, identified by heuristics measuring the entropy of measurement probabilities [95],[96]. However these heuristics assume low entropy correlates to informative observations, which is not necessarily true in general.

Alternatively, information gathering can be promoted by action design, such as by providing high reward when taking a specific action in a state of interest [97], though this approach requires an action for each such state. Other work provides online performance for problems where part of the state space can be observed directly [98]. This approach has also been extended to continuous settings by augmenting the reward with convex information measures on the belief-space [99] which includes the information gathering reward we consider in our work.

Control Based Safety

While active fault estimation can more rapidly determine the failure of a robot, many systems have operational safety constraints which the information gathering actions must not violate. Traditional approaches to ensuring safe control include formal methods such as control barrier functions (CBFs) [56], and their extensions to discrete time systems [48] with stochastic noise [52]. In the face of randomly changing environmental hazards, metrics to quantify the risk a robot faces can be used to plan safe trajectories, such as using entropic value at risk to bound tail probabilities [100]. Other planning-based approaches to real-time optimal control with safety constraints include Sequential Convex Programming (SCP) [49], [50] which can consider complex [101] and stochastic constraints [51] while achieving robustness and stability guarantees through tracking control [102].

However, each of these methods assumes a fully observable state to directly evaluate safety constraints. In partially observable settings, these methods must be modified, such as by extending CBFs to operate on the belief of possible states [103] using particle filters, and the conditional value at risk, an alternative risk adverse bound on tail probabilities. Partial observability can also occur from uncertain system dynamics, such as the unknown failures we consider. When feasible, applying a CBF to all possible dynamic modes can ensure safety, but this approach may be overly conservative [104], leading to short or no horizons with guaranteed safety. Our method extends this consideration of multiple possible system dynamics by also considering how the planned actions will help distinguish between the different possibilities, branching on this information to achieve longer safe horizons.

In comparison with other methods, these control approaches are typically fast to execute and can handle complicated safety constraints, as we visualize in Fig. 4.1. However, CBFs and SCP typically only impose deterministic constraints on the system. They are also limited by the inability to consider multiple system models and information gain from actions, which we observe limits their performance in our experiments.

Our work sits at the intersection of these several related fields as we seek to combine these separate approaches of fault estimation and constraint satisfaction. We qualitatively summarize this relation in Fig.4.1. Like the passive fault estimation and POMDP approaches, our algorithm is applicable to a wide range of stochastic and uncertain systems. Similar to control-based safety methods and POMDP models, our method, with mild assumptions, provides formal guarantees of constraint satisfaction and general bounds on tail probabilities as opposed to constraints on expectations alone. However, unlike existing active fault diagnosis methods or POMDP solvers, our method can be deployed online in information gathering problems without requiring heuristics.

4.2 Significance of s-FEAST

The experiments presented in the last two chapters have demonstrated safety-aware, real-time, active fault estimation that extends beyond the capability of existing methods which cannot succeed at all tasks simultaneously. Indeed, we have shown that in the presence of ambiguous faults and time-critical constraints, proactively taking actions to maintain safety while simultaneously gathering information about the system status is necessary to reliably avoid collision. By combining anytime tree search with efficient filtering and probabilistic chance constraints, s-FEAST achieves both objectives in a computationally tractable fashion, with asymptotic convergence guarantees and interpretable behavior.

Comparison with Existing Tree-Search Methods

Previous approaches to solving partially observable planning problems using tree-search methods include the Partially Observable Monte Carlo Planning (POMCP) algorithm [37] and its extensions to constrained systems [85], [86]. However, in Chapter 2 we empirically showed that our marginalized filter approach is necessary for effective planning in information gathering problems. When the reward is a function of the belief instead of just the classical state and action reward, the convergence guarantees of these existing methods breaks down. In this section, we formalize this observation.

POMCP consists of two components. First, Partially Observable Upper Confidence Bound applied to Trees (PO-UCT), which assumes access to the state belief for a given history, and second, Monte-Carlo updates to propagate the belief within the tree in a particle filter like manner. For each simulation, a particle is sampled from the initial belief, and propagated by running PO-UCT. At each belief node encountered during the simulation, the propagated state is added to the node's particle belief. At each node, the resulting belief is a discrete collection of state particles, one for each visit to the node.

POMCP argues that at a large number of samples, the belief at each node is well approximated such that PO-UCT is solving the equivalent Belief Markov Decision Process (Definition 5) and therefore inherits the value convergence of the fully observable UCT [35]. However, it only establishes this for the PO-UCT algorithm, as the theoretical analysis assumes accurate state beliefs for each history and accurate rewards for each node. Neither is initially true in the information gathering setting, where estimation is inherently coupled with the reward. This leads to a "burn in" phase until the belief converges enough that this PO-UCT analysis is valid. In fact, until a repeated particle is added to a node, the information gathering reward we introduced in Chapter 2 (Eq. (2.20)) is inversely correlated to the number of visits to the node. This results in a breadth first search where the UCT strategy of biasing towards areas of high reward no longer succeeds, and ultimately random action selection. This is further exasperated by the exponential scaling of standard particle filters with the number of dimensions [105]. The difference in tree growth between s-FEAST and POMCP is visualized qualitatively in Fig. 4.2.

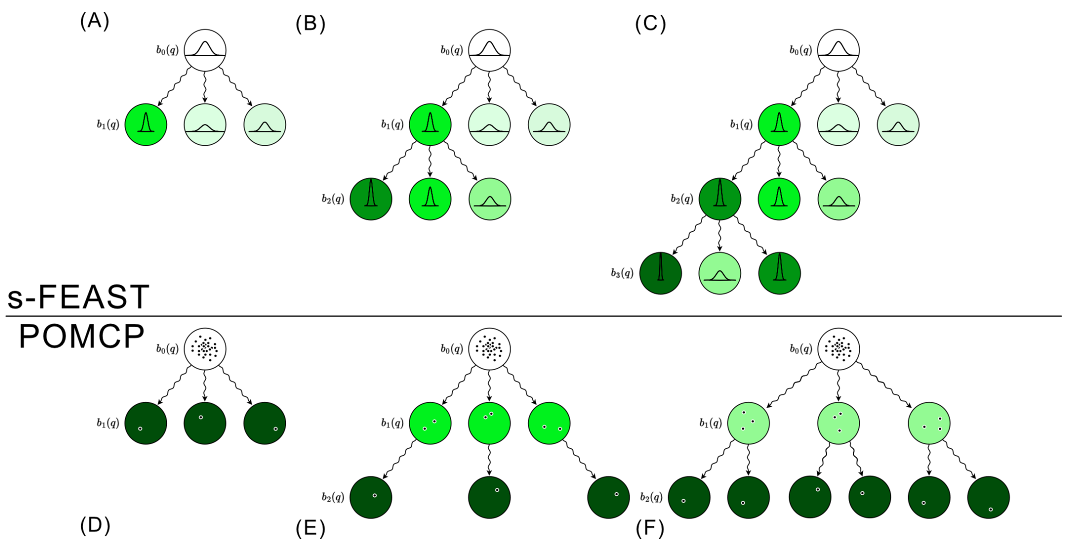


Figure 4.2: **Marginalized filtering vs. particle filtering in information gathering tree searches.** A conceptual comparison of the tree growth of s-FEAST (which uses our marginalized filter) and POMCP (which uses a particle filter). Darker green is a higher estimated reward. (**A**) When s-FEAST expands a node, it performs full belief updates that give accurate reward estimates. (**B**) The tree can then be biased towards areas of higher rewards. (**C**) This enables s-FEAST to efficiently search areas of higher value and plan further ahead. (**D**) In comparison, POMCP performs a particle filter-based search, adding one particle each time it visits a node. When POMCP has only encountered a node once, the estimated reward given by Eq. (2.20) is maximized. However, this estimate is inaccurate. (**E**) As soon as POMCP re-visits a node, there are now two particles, leading to increased belief uncertainty and less reward. (**F**) The result is a breadth first search.

Real-Time Performance

Our solver is an anytime algorithm, which means its performance improves given more computation time, but it can be stopped at any point to return the current best solution. In our real-time hardware experiments, the solver evaluation is not fast enough to achieve the highest $N = 2000$ level of planning we consider in Fig. 3.3. Instead, when running s-FEAST on the robotic spacecraft simulator, we typically evaluate $N = 85$ trajectories, which is sufficient to successfully identify faults and maintain safety while substantially outperforming baseline methods. This achieves the goal of validating our conceptual algorithmic innovations, although it is possible to further optimize the software and hardware implementation for a faster run time and better performance.

To this end, we note s-FEAST presents several promising opportunities for future performance improvements. First, the marginalized filter we presented in Chapter 2 factors out the physical state estimators, meaning that s-FEAST can leverage any existing estimators that may already be optimized for a system with minimal changes. Second, there exists a growing body of literature on methods to accelerate partially observable planning through parallelization [106] and GPU use [107]. Finally, in the next chapter we will explore an alternative array-based implementation aimed at optimizing tree search performance on pipelined processors. We view these types of optimizations as complementary to s-FEAST's algorithmic innovations which achieve better scaling through exploitation of the active sensing problem structure. They also pair well with the anytime nature of our algorithm, as increasing simulation speed directly translates into more simulations and improved performance, as seen in Fig. 3.3 (B and D).

The anytime property is also desirable compared to traditional active fault diagnosis methods, which often require the computation to complete before a solution can be returned and may employ approximations to achieve real-time performance [73]. Instead, we can return the best solution found within any computation budget, and we have shown in Chapter 3 that s-FEAST converges asymptotically to the optimal solution, a guarantee not provided by existing chance constrained anytime methods [89].

We expect the ongoing trend of ever-increasing computational power onboard space robotics missions [13], [14] to be enabling for our methodology, especially as payloads are developed for increasingly data-intensive science applications.

Envisioned Use Cases

As our algorithm only needs to run when a fault is suspected or a safety-critical situation is encountered, we envision a concept of operations where our algorithm is dormant until needed, in which case it takes priority over non-essential payload operations to monopolize computing resources for a short duration, before handing back control when normal operations can resume. Our algorithm could also run at scheduled intervals to proactively check for possible faults, resulting in planned payload down time much like other maintenance operations including charging windows and course corrections that mission planners currently consider.

This intermittent approach will also minimize the fuel or other critical resources used by s-FEAST. If proactive runs are scheduled during mission formulation, s-FEAST's resource use could be explicitly budgeted for. Further, this budget, or fractions of it, could be formulated as another constraint that s-FEAST must satisfy over its planning horizon in addition to safety. This approach could explicitly balance minimizing short term dangers with the need to preserve long term capabilities for mission life time, much like approaches balancing potential science payoffs with the risk of mission failure [108].

Our algorithm will be most effective in systems with high functional redundancy where faults must be diagnosed in a matter of minutes. These scenarios are likely to create ambiguity between possible failures, which s-FEAST can resolve. The time frame also precludes ground intervention and requires autonomous capabilities, but still allows enough time to take test actions onboard the system and gather information. In particular, we envision s-FEAST enabling missions to accept higher levels of risk, stemming from lower budgets, less preflight qualification, or harsher environments, by providing a reliable method to identify faults when they occur, enabling effective recovery by the onboard controller.

When fault recovery must occur in a manner of seconds, there is not enough time for s-FEAST to gather additional information. If ambiguity exists between possible faults, recovery options such as switching to a fully redundant back up system may instead be necessary as opposed to determining the loss of capabilities and adjusting the controller accordingly. Here however, our marginalized filter may still provide an effective method to distinguish between multiple fault scenarios by combining prior failure likelihoods with the available observations in an efficient and optimal manner. Similarly, when fuel budgets are particularly tight, we can run our marginalized filter passively to monitor the likelihood of dangerous faults, only taking actions to intervene or gather further information when the safety of the system drops below a tolerable threshold.

Application of s-FEAST to Other Information Gathering Problems

We note that our approach to belief-space planning and sampling-based safety can be applied to other information-gathering problems where the underlying state has a tractable belief-update.

For example, we can consider the classic problem of a robot autonomously mapping an unknown environment [109] or future robotic planetary exploration missions where actions are taken to scout out areas of potentially high scientific value [110]. In both cases, gathering information is a key goal, necessitating belief-space planning. And in both cases, the robot might be subject to additional requirements where the effect of high variance makes constraints on expectation alone limiting; ranging from the safety constraints we consider here to battery or time budgets limiting exploration.

Limitations

While we believe s-FEAST provides a strong mix of real time performance and generality, it is not without limitations. As a model-based approach to fault identification, s-FEAST's performance is dependent on the accuracy of the dynamics, measurement, and fault models. Modeling error can be accommodated by increasing the noise or adding unknown disturbance terms. However, these approaches decrease the predictive power of our marginalized filter and tree search. In our hardware experiments, we adopted a conservative noise term to account for errors in our onboard model, but found that we needed to calibrate for residual friction terms to achieve reliable performance.

Another limitation of our method is the need to enumerate in advance the faults that s-FEAST will consider. While it is often the case that most likely or risky failure modes can be determined through modeling, ground testing, or operational history, this limits our ability to autonomously handle previously unseen and unconsidered faults.

To address this, opportunities for future improvements of s-FEAST include broadening the current enumerated sets of discrete possible faults to bounded subspaces. This would allow for scenarios where the faults of concern are not known in advance, but the bounds on possible fault behavior are. Another avenue for future work would be to unify s-FEAST with data-driven approaches for safe exploration of unknown, fully observable dynamics to actively estimate unmodeled actuator or sensor faults [111], [112]. Similar methods of learning residual dynamics online could also be employed to mitigate the effects of modeling errors [113], [114].

Finally, we note that our method will fail if a safety-critical state of the system becomes completely unobservable or uncontrollable. However, these situations will be unrecoverable for our baselines and related work as well.

4.3 Chapter Summary

In comparison with other approaches to planning for safety and information gathering, s-FEAST provides a flexible method for online active fault estimation in safety-critical settings. Through the use of an efficient marginalized filter, s-FEAST can perform information gathering in settings intractable to existing tree-based solvers.

Its modular nature presents opportunities for future performance enhancements of s-FEAST by estimator or tree search optimizations. Further, its anytime property allows s-FEAST to scale its performance with the available computational budget.

Chapter 5

AN ARRAY-BASED IMPLEMENTATION OF MONTE CARLO TREE SEARCH

5.1 Motivation

In the pervious chapters, we saw that both FEAST and s-FEAST's performance heavily depended on how many iterations of the tree search could be run within a fixed computational time. In particular, we were only able to reliably run s-FEAST at a rate of about 85 simulations per tree in our robotic experiments, which was just barely enough to achieve the results we presented.

Given that we saw improvements up to simulation levels of at least $N = 2000$ in Fig. 3.3, there is significant unrealized potential that can be gained by either increased computational capabilities, like we discussed in the previous chapter, or algorithmic innovations which lead to more efficient searches. Heuristic approaches are also possible, and we discuss them at the end of the of the chapter. However, these methods suffer from a loss of the theoretical guarantees we used to prove the convergence of FEAST and s-FEAST in Chapters 2 and 3.

This motivates us to consider methods of accelerating tree searches without changing the underlying algorithm. One promising method would be to employ hardware accelerators, such as GPUs. While these have presented promising results in the acceleration of rollouts to terminal states and other readily parallelizable aspects of tree searches [107], [115], [116], deploying a full tree search algorithm on a GPU is limited to due the high memory latency and limited branching performance [117]. In our own development of FEAST and s-FEAST, we encountered similar constraints when using software packages that compiled code into fast, low level instructions, but needed to return to higher level and lower performance code whenever branching occurred [118].

Branches are a type of control flow commonly found in programs where the execution path taken depends (or branches) on the result of a previous instruction. One of the most common occurrences of branching is an "if" statement, which splits execution based on a boolean value. The reason branches result in decreased computational performance is due to the pipelined architecture that is nearly ubiquitous on modern processors and is visualized in Fig. 5.1. To leverage the full performance of a CPU

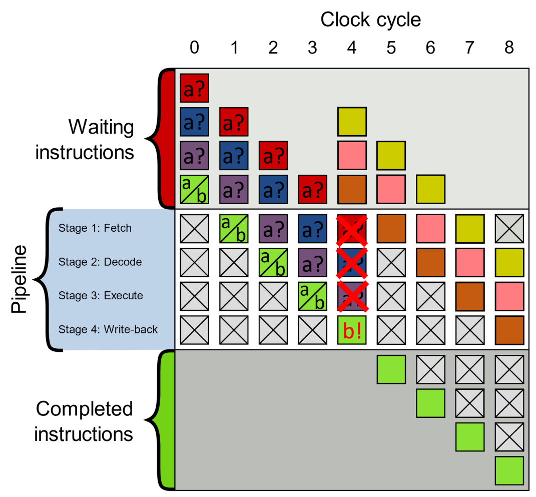


Figure 5.1: **Processor Pipelining.** A pipelined processor carries out instructions (represented as colored blocks) in stages. Here, each column represents a processor clock cycle. Between clock cycles, instructions are advanced through the pipeline, allowing each component of the hardware to be utilized simultaneously. When branching between multiple execution paths occurs, the processor guesses which instructions to load. If it guesses incorrectly, the instructions must be flushed and replaced with the correct execution path. [1]

(or GPU), instructions (such as adding two numbers) are queued and processed in order through different sections of the processor simultaneously. When a branch occurs, the processor must guess which execution path will be followed next, and load the appropriate instructions. If the guess is incorrect, these instructions must be flushed, and new instructions loaded at the start of the pipeline, resulting in lost computation cycles [119]. When some instructions or data must also be fetched from memory, this effect is further exasperated by memory latency [120]. The combined effect is that choice of algorithm architecture can have significant effects when deployed to real systems, leading to the development of techniques to avoid expensive instructions [121].

Branching is fundamental to tree searches, as we need to decide which child node to proceed to (or new child to initialize), at each level of the tree, for each simulation. In the case of FEAST and s-FEAST, we perform expensive dynamics and belief updates whenever a new node is generated. For a tree with $N = 2000$ simulations, 40 possible failures, and a max depth of 4, we compute at worst case 320,000 filter updates. These are primarily dominated by matrix operations, and would be well

[1]Adapted from "A generic 4-stage pipeline" by Cburnett under `https://creativecommons.org/licenses/by-sa/3.0/`

suited to parallelization on a GPU or other accelerator. Unfortunately, because of the branching at each level of the tree, we can only consolidate the operations into 8,000 batches, one for each branch we take.

This motivates us to consider implementation improvements to UCT (and PO-UCT) [35], [37] that make the algorithms more efficient on computing hardware. In particular, we are interested in developing an implementation with predictable branching that may be better suited to accelerators such as GPUs.

5.2 Algorithm Overview

Our objective is to construct a version of MCTS with a predictable execution path by using arrays as the primary data structure. This will allow for better processor performance, by avoiding branch prediction failures in the processor pipeline and by leveraging existing optimizations for array operations.

Our method is illustrated by classic techniques to avoid branching instructions when deciding between two values depending on a boolean variable. Instead of using an "if" statement, the answer can be computed using arithmetic or binary operations [122]. This form of micro optimization is valuable on processors which feature expensive branching, have limited access to compiler optimizations, or lack conditional move instructions. An example is shown below in in Algorithm 4 for a system with 8-bit integers.

Algorithm 4: Three equivalent implementations of selecting the integer a or b based on the value of the condition c. The first implementation introduces a branch, the last two do not.

```
int8 a ; int8 b ; bool c ; int8 r ;
if c then
    r = a;
else
    r= b;
/* Equivalent Implementation                              */
r = a * c + b * !c ;
/* Equivalent Implementation                              */
int8 m = c ;
m= m | m « 1;
m = m | m « 2;
m = m | m « 4;
r = a & m| b & m ;
```

In MCTS, the fundamental instance of branching is deciding whether to add a child node or not. If the selected action (or state), has already been added to the tree, then the previous node is retrieved and the algorithm traverses downwards. If it has not been, then a new node must be initialized before continuing. Unfortunately, this occurrence of branching is more complicated than the example of Algorithm 4, as the behaviors are fundamentally different.

However, this example still gives a clue to how the branching can be made predictable. If we considered the child nodes to be arranged in a list ordered by time of insertion as diagrammed in Fig. 5.2, then matching an existing node is equivalent to finding the index of the matching child within this list. Conversely, failing to match a node could be represented by the index past the end of the list. Following the example in Algorithm 4, the index of the matching child is our first integer, `a`, the index of the new node is the second integer, `b`, and the boolean condition, `c`, is now whether or not a match occurred. We note that while the value of the match index, a, may be ill defined if no match exists, because its value will be discarded, this is no issue.

We have now made selecting a child predictable, but we run into one more challenge. When adding a new node to the tree, we may need to perform some initialization steps, or otherwise execute different behavior than when selecting an existing node. It is still possible to avoid guessing which execution path to follow, but it now comes at a price. Instead of performing one set of behavior conditioned on whether the node is a new or existing child, we perform both behaviors every time, and use techniques like that of Algorithm 4 to nullify the operations that should not be performed. This is the trade off for predictable execution. To always know in advance which instructions will be executed, we now must perform some repeated and redundant computation. However, we will show that the resulting optimization for pipelined processors outweighs this penalty.

This discussion outlines the general approach we take when developing our array-based MCTS algorithm. In the next section, we will focus in detail on developing the algorithm for discretized dynamical systems like we considered in Chapters 2 and 3. However, we note that our algorithm will also work in other systems that can be formulated as an MDP or POMDP, and is in fact simplified in systems with bounded state transitions.

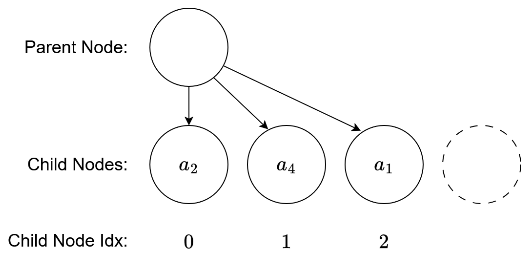


Figure 5.2: **Child node diagram.** The child nodes can be thought of as an indexed list. In this case, taking action a_4 would be equivalent to selecting index 1, whereas taking action a_0 would be represented by index 3, indicating that child has not yet been added to the list.

5.3 Array-Based MCTS Algorithm

MDP Development

We first develop our array-based tree search algorithm for fully observable MDP systems, to build intuition and validate our performance. We will then extend to POMDP systems, including the noisy planar spacecraft system we validated FEAST and s-FEAST on in the next subsection.

Much like in our development of FEAST, fully observable dynamical systems have two types of nodes within the tree search, action nodes and state nodes, which together make up a single layer of the tree, as shown in Fig. 5.3 for a simple system with three actions and three states. Layer zero is degenerate, with no actions and only the root state. Each node stores the total number of visits so far, the value or reward at the node, and the node's children. To simulate down the tree, an action is selected according to the UCT augmented value function:

$$\hat{V}_{\text{aug}}(H \cup a) = \hat{V}(H \cup a) + c\sqrt{\frac{\log N(H)}{N(H \cup a)}} \tag{5.1}$$

where like in FEAST and s-FEAST, H represents the history in the tree up until the current state node, and $H \cup a$ represents taking action a at this point in the tree. $\hat{V}(\cdot)$ and $N(\cdot)$ represent the value estimates and visit counts for given histories, respectively. The next state is generated by simulating the system forward using the selected action. In continuous systems, the resulting state is discretized and compared to all the current child state nodes. If no match is found, a new child is initialized. The search continuous until the simulation depth has been reached, then backpropagates up the tree, adjusting the value and visit counts as it goes.

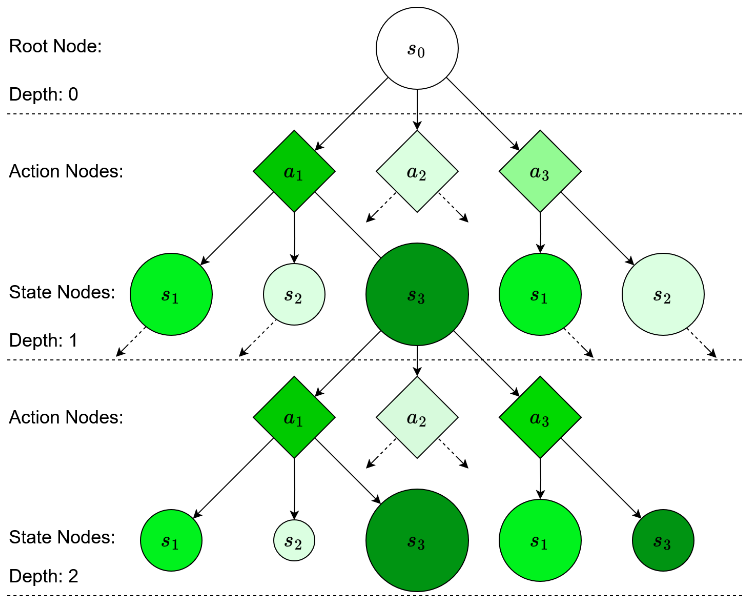


Figure 5.3: **MDP tree search diagram.** Diagram of a tree search over a simple MDP with 3 actions and 3 states. The tree has been split into depth layers, with depth 0 containing only the root node. Node size represents relative visit frequency. Darker colored states represent higher rewards, and darker action nodes represent higher values resulting from the weighted average of the rewards encountered under them. Note that each action may not result in all possible states.

Typically, this algorithm is implemented using tree data structures. These hold a reference to the parent node, the current value or reward, visit count, and a list of each child action or state node. While the number of possible actions is fixed, the number of states may not be, or may be very large. This is particularly true in dynamical systems with Gaussian noise models like the planar spacecraft model we consider. A strength of tree data structure implementations is that this branching is handled naturally by storing an extendable list of children nodes. However, by nature of searching over an unknown problem, the branching encountered throughout the tree is hard to predict.

Instead, in our algorithm, we store each action and state node within a layer in two shared arrays. Now, adding a child to a node means adding a new node to the end of the appropriate array. Rather than directly containing their children, a node now saves the index of each child. This modified structure is visualized in Fig 5.4 for the first six simulations of the simple three action and three state system we considered earlier. The nodes in each layer are now ordered by time of creation, with the children of different parents being intermixed.

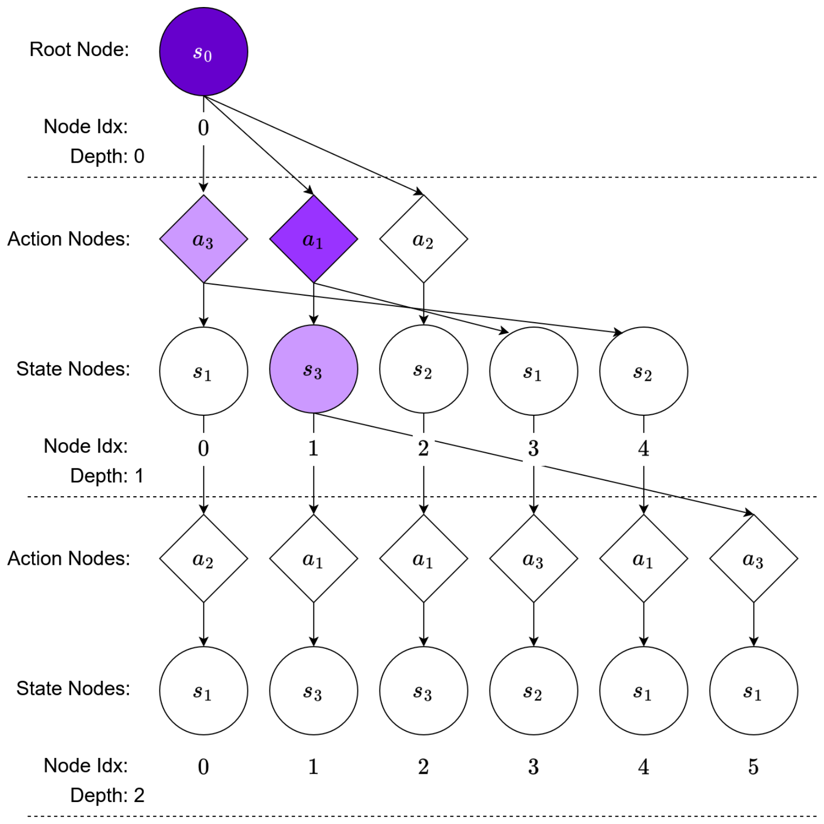


Figure 5.4: **MDP node array diagram.** Diagram of the first six simulations of the same tree search as shown in Fig. 5.3. Each node is now sorted by type within a layer and assigned an index when added. Darker colors represent the visit frequency of each node. The values and rewards are not visualized here.

One observation that can be made immediately is that the length of each node array increases by layer. At depth 1, the action nodes are fully expanded, as all three actions which can be taken from the root node have been tried. Conversely, four more state nodes can be added to the first layer to reach a total of nine, three for each of the three actions. These nine state nodes can themselves have three children, and so on. This observation that the branching is bounded, if growing geometrically by layer, motivates our decision to consider a hierarchical structure where the data from each layer is stored in separate arrays, as opposed to storing all state and all action nodes within the same array. By separating out the smaller layers near the root node, we can efficiently store them and increase the likelihood they are kept within the processor cache, leading to lower latency and faster execution. We will show in our numerical results that this method leads to a substantial speed up at the trade off of somewhat a more complicated data structure.

We have established the architecture for organizing the nodes into layers, and this can be done to store the parents and visits to each node, as well as the states and action values. However, one challenge remains in how to effectively store the child indexes while predictably determining whether or not a new child needs to be created. In the case of the action node children of a state node, we can take advantage of the fact that the set of available actions is constant and numbered throughout the tree search. Therefore, we consider an array of child indexes for each individual belief node, resulting in a 2D array for each layer. Each column corresponds to a single state node, and each row represents a consistent action, eliminating the need to store the action taken within the child node. Our action selection method is shown in Algorithm 5 below. We are given the current depth and state index, and begin by retrieving the indexes of each child action. If unassigned actions exist, these entries are initialized to a value one greater than the maximum possible index. This corresponds a virtual action which is never visited, ensuring we try the unassigned actions. If none exist, we set `untriedAction` to false and retry the action with the best UCT value.

Algorithm 5: Predictable Action Selection.

```
def selectChildAction(depth, curStateIdx):
    childActionIdxs =
      childActionNodes(depth).column(curStateIdx) ;
    childActionValues = actionValues(depth + 1)[childActionIdxs] ;
    childActionVisits = actionVisits(depth + 1)[childActionIdxs] ;
    bool untriedAction = anyUnvistedChild(childActionVisits) ;
    uctValues = uctValue(childActionValues,
      childActionVisits,stateVisits(depth)[curStateIdx]) ;
    int8 bestAction = maxIndex(uctValues) ;
    int8 newAction = getRandomUntriedAction(childActionVisits) ;
    int8 nextAction = newAction * untriedAction + bestAction *
      !untriedAction ;
    int8 nextActionIdx = numActionsAtDepth(depth + 1) *
      untriedAction +
      childActionNodes(depth)[bestAction,curStateIdx] *
      !untriedAction ;
    childActionNodes(depth)[nextAction,curStateIdx] =
      nextActionIdx ;
    numActionsAtDepth(depth + 1) += untriedAction ;
    return nextAction ;
```

where `depth` and `curStateIdx` represent the current position of the tree search, `actionValues`, `actionVisits`, `stateVisits`, `numActionsAtDepth`, and `childActionNodes` represent global tree data arrays we can access by layer, and we use the helper functions anyUnvistedChild, uctValue, getRandomUntriedAction to determine if unvisited child actions exist, compute the UCT value function (Eq. (5.1)) for each action, and return a random untried action (or a default value if none exist), respectively. Note we only need to return the selected `nextAction`, as all other information has been stored in the global tree data arrays.

We use a similar method for the child state nodes. However, this is complicated by the fact that the states are not already ordered like the actions are, and may have unbounded branching as discussed before. To address this, we make two modifications to the child state node array from the architecture of the child action node array. First, we now add the state nodes to each column in order of generation, instead of assigning a consistent meaning to each row. To track the number of state nodes already added to each column (and hence where to add a new child state node), we add a row to the bottom of the array which counts the number of state nodes already present, initialized to zero. This doubles as the index which a new state should be added to.

The other change we make from the child action node array is to assume a max level of state branching. In simple MDPs, way may know this branching exactly, but in more complicated problems it may be unclear. In the case of our planar spacecraft with Gaussian noise, the branching is actually unbounded, as outlier states are rare but possible. However, we also know that for each action taken, the resulting states will be clustered around the noise free dynamics. For a given number of simulations in the tree and noise level, we can assume a maximum level of branching and verify it empirically. Further, we can also set different levels of assumed branching by search layer, taking advantage of the our knowledge that actions closer to the root will be taken more often, leading to more chances of branching to outlier states, and actions further down the tree will branch less often. Other approaches could be taken to bound the amount of state branching, including clipping the noise distribution or varying the amount of noise or discretization in different layers of the tree.

The resulting child state node selection algorithm is given in Algorithm 6 below. We are given the current depth, action index, and the state generated by simulating the selected action. Like when selecting a child action, we begin by retrieving the first $N_{S,l}$ child state indexes from the column of the current action node, where $N_{S,l}$ is the assumed state branching at layer l. If unassigned states exist, these entries are initialized to point to the last possible state in the layer, which has a value initialized to NaN. This ensures these entries will not match when we compare each child state against the given state. We set the `matchIdx` to the child state matching the most elements of the given state, however the `matchFlag` is only set if all states match. As with the action selection, we use this flag to distinguish between whether to write to `matchIdx` or a new entry in the child state node array, as well as determine the value of `nextStateIdx` and if we should update the number of state nodes in the layer and the current column of the child state node array.

Algorithm 6: Predictable Child State Selection.

```
def selectChildState(depth, curActionIdx, generatedState):
    childStateIdxs = childStateNodes(depth).column(curActionIdx);
    childStates = stateNodes(depth)[childStateIdxs] ;
    for $i = 1 \ldots N_{S,l}$ do
        stateMatches[i] = Σ (childStates[i] == generatedState) ;
    int8 matchIdx = maxIndex(stateMatches) ;
    bool matchFlag = stateMatches[matchIdx] ==
      size(generatedState);
    int8 idxInChildArray = matchIdx * matchFlag +
      childStateNodes(depth)[ $N_{S,l}$,curActionIdx] * !untriedAction ;
    int8 nextStateIdx =
      childStateNodes(depth)[matchIdx,curStateIdx] * matchFlag +
      numStatesAtDepth(depth) * !matchFlag ;
    childStateNodes(depth)[nextStateIdx,curActionIdx] =
      nextStateIdx ;
    stateNodes(depth)[nextStateIdx] = generatedState ;
    childStateNodes(depth)[$N_{S,l}$,curActionIdx] += !matchFlag ;
    numStatesAtDepth(depth) += !matchFlag ;
    return nextStateIdx ;
```

where `curStateIdx` is the index of the action node we are deteriming the child of, `generatedState` is our generated state to compare against, `stateNodes`, `numStatesAtDepth`, and `childStateNodes` represent global tree data arrays we can access by layer, Σ sums the number of matched states, maxIndex() returns the index of the best matching child state, and size() returns the number of elements in the state. There is no issue with non-unique maxima for `stateMatches` as this only occurs when no exact matches exist, resulting in `matchFlag` being set to false. Again we only need to return the selected `nextStateIdx`. All other information has been stored in the global tree data arrays.

Combining these two components results in our full array-based MCTS algorithm, visualized below in Fig. 5.5. Each layer consists of 8 arrays, divided between the action and state nodes, with columns representing a single action or state within a layer. The parent state and action arrays store each node's parent index. Alternatives to Algorithms 5 and 6 can be constructed by searching over the parent arrays, however these require exhaustive searches that scale poorly with the number of nodes in a layer, and return varying numbers of nodes, which could reintroduce the need for branch prediction.

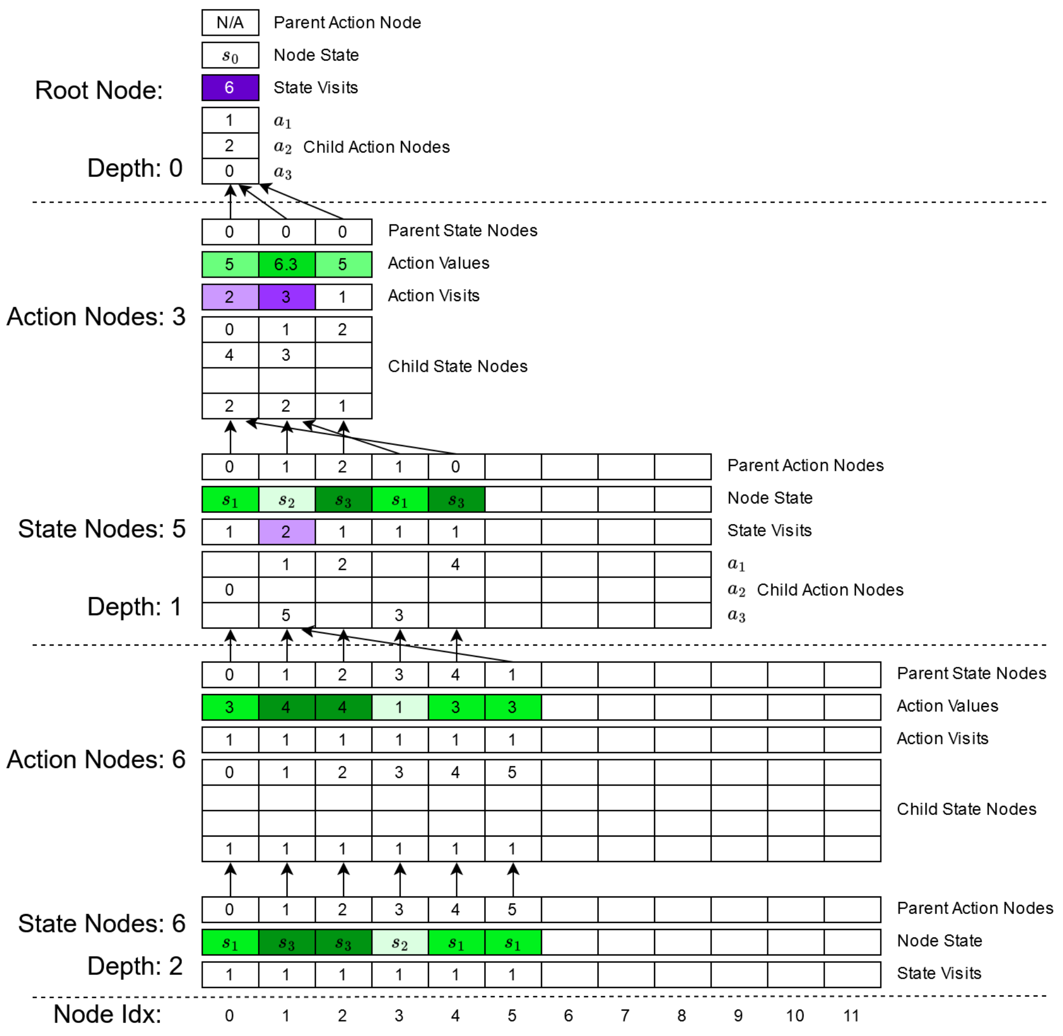


Figure 5.5: **Array-Based MCTS Diagram.** Diagram of the first six simulations of of the same tree search as shown in Figs. 5.3 and 5.4, now using an array-based MCTS representation. Each layer consists of 8 arrays, 4 each corresponding to action and state nodes. Each column corresponds by index to a state or action node within the layer. The parent array gives the indexes of each node's parent shown by the arrows. The child action arrays store the indexes of the child actions in the layer below for each available action, if any. The row of the child index indicates the action taken (note the actions are 1-indexed). The child state arrays point to the child state nodes that have resulted from a given action. The states nodes are sorted by order of creation, with the last element each column indicating what index the next child should be added to. The columns in layers 0 and 1 are limited by the maximum possible branching due to the number of actions and states. The number of columns in layer 2 is limited for visualization. For clarity, array entries which have not been written to during the search are left blank. Darker colored states represent higher rewards, and darker action values represent higher weighted averages of the rewards encountered after the taken action. Darker visit counts represent a higher frequency of visiting each node. The final layer has no child actions.

We present the complete pseudocode of our array-based MCTS algorithm below in Algorithm 7. We first initialize the data arrays for each layer of the search to the default values of zero, unless indicated otherwise. We then iterated down the tree, using Algorithms 5 and 6 to select the action and state child nodes before backpropagating up the tree in the standard MCTS fashion. This process is repeated the specific number of simulations, and the action with the best value is then returned.

In Algorithm 7, `rootState` is the initial state, N the number of simulations to perform, `maxDepth` indicates how many layers are present in the tree, `actionSet` is the fixed set of actions for all layers, $N_{S,1-\texttt{maxDepth}}$ represents the state branching limits at each depth, const(m,n,c) is a function which creates an m by n matrix with each entry set to c, and rewardFunction() gives a reward based on the current state. The variables `curMaxNodes` and `curMaxChildNodes` capture the maximum possible branching at the current layer and its children. Knowing that we start with a single root node, there are at most size(`actionSet`) actions below it, and size(`actionSet`)$*N_{S,1}$ state nodes below that, and so on. We also know that at most one action and one state node can be added per layer, per simulation, leading to an alternative bound on the number of nodes per layer. This limit on the number of states per layer can be seen visually in Fig. 5.5. Finally, we note that there is no discounting in the backpropagation step in Algorithm 7, as we consider the finite horizon setting, but it can easily be added in.

POMDP Development

Extending our array-based MCTS search to apply to POMDPs in addition to fully observable MDPs is straightforward. Since the true state of the system is now unobservable to the tree solver, we are instead given an initial belief at the root node, and propagate this forward in the tree search, replacing state nodes with belief nodes. A belief node differs from the state nodes used in the MDP implementation in two ways. First, instead of storing and comparing simulated states to determine if a new child node should be appended under an action node, we use the observations generated.

The second change is that we also store the belief at each node, by propagating it forward for a given action and observation pair. For example, to adapt FEAST and s-FEAST to an array-based implementation, we would use the marginalized filter to propagate the belief, and the beliefs in each layer would be stored in 4 dimensional array, for each node, failure, and two dimensional covariance. This is

Algorithm 7: Array-Based MCTS

```
def arrayBasedMCTS(rootState,N,maxDepth,actionSet,$N_{S,1-maxDepth}$):
    /* Initializing the Root Node */
    stateNodes(0) = rootState;
    childActionNodes(0) = const(size(actionSet),1,size(actionSet)) ;
    /* Initializing Tree Data Arrays at Each Depth */
    curMaxChildNodes = size(actionSet);
    for $l = 1 \ldots maxDepth$ do
        curMaxNodes = curMaxChildNodes;
        curMaxChildNodes = min($N$,curMaxChildNodes*$N_{S,l}$);
        childStateNodes($l$) = const($N_{S,l} + 1$,curMaxNodes,curMaxChildNodes-1)
          ;
        childStateNodes($l$)[$N_{S,l}$,:] = 0;
        curMaxNodes = curMaxChildNodes;
        stateNodes($l$)[curMaxNodes-1] = NaN ;
        curMaxChildNodes = min($N$,curMaxChildNodes*size(actionSet));
        childActionNodes($l$) =
          const(size(actionSet),curMaxNodes,curMaxChildNodes) ;
    for $i = 1 \ldots N$ do
        curStateIdx = 0 ;
        curState = rootState ;
        /* Simulate Down the Tree */
        for $l = 0 \ldots maxDepth - 1$ do
            curAction = selectChildAction($l$,curStateIdx) ;
            curActionIdx = childActionNodes($l$)[curAction,curStateIdx];
            stateParentNodes($l + 1$)[curActionIdx] = curStateIdx;
            curState = simulate(curState,actionSet[curAction]);
            curStateIdx = selectChildState($l + 1$,curActionIdx,curState);
            actionParentNodes($l + 1$)[curStateIdx] = curActionIdx;
        summedReward = 0;
        /* Backpropagate Up the Tree */
        for $l = maxDepth - 1 \ldots 0$ do
            summedReward += rewardFunction(stateNodes($l + 1$)[curStateIdx];
            curActionIdx = actionParentNodes($l + 1$)[curStateIdx];
            actionVisits($l + 1$)[curActionIdx] += 1;
            actionValues($l + 1$)[curActionIdx] += (summedReward
              -actionValues($l + 1$)[curActionIdx] )/
              actionVisits($l + 1$)[curActionIdx] ;
            curStateIdx = stateParentNodes($l + 1$)[curActionIdx];
            stateVisits($l$)[curStateIdx] += 1
    return arg max actionValues(0) ;
```

expensive, as is recomputing the belief update each time we regenerate a belief node to avoid branching. In the chapter summary, we discuss possible hybrid approaches to mitigate the price paid for excess repeated computation.

5.4 Preliminary Results

Having developed our array-based MCTS algorithm in the pervious section, we analyze its performance in both MDP and POMDP settings here. We compare against baselines of a traditional tree-based implementation and an alternative architecture which uses a single state node array and single action node array shared by all layers of the tree, to justify our choice of a layer based approach. For consistency, each search method was implemented in C++17, with the same compiler optimizations and libraries. All experiments were performed on a AMD Ryzen 7 3700X (8-Core Processor, 3593 Mhz) with 32 GB of memory available.

First, we validate that our array-based MCTS can solve planning problems by considering a MDP with a non-convex obstacle. The challenge is that planning methods must be able to consider far enough into the future to discover that going around the obstacle is optimal, as opposed to a simple gradient based solution which will get stuck against the obstacle wall. We used the same planar spacecraft dynamics model as in Chapters 2 and 3 to validate FEAST and s-FEAST, except that the model is now fully observable and not subject to any faults. In Fig. 5.6(A), we present a 12 time step simulation, demonstrating our array-based method can successfully solve this MDP problem. In Fig. 5.6(B) we show the growth of the array-based search over a single time step, validating that a branching search is being performed. Note that the state discretization results in branches snapping to the same points in space. Some simulations result in points inside the obstacle. These receive a large negative penalty, and are not selected by the algorithm.

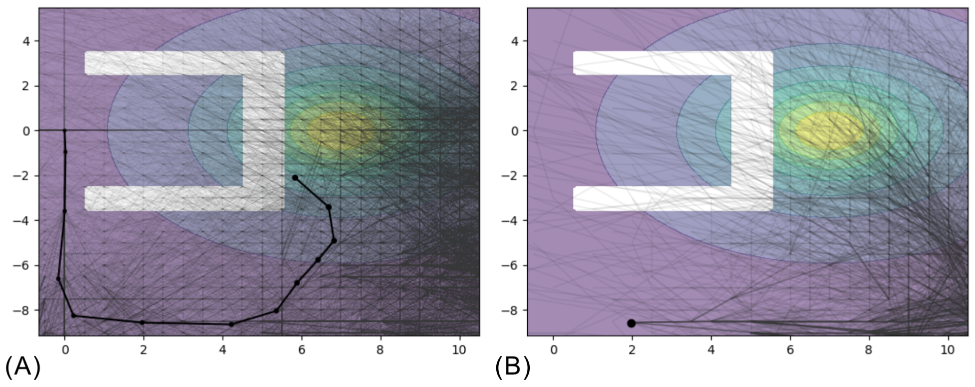


Figure 5.6: **Array-based MCTS solving a non-convex problem.** Tree search around a non-convex obstacle. (**A**) The array-based MCTS method successfully escapes the local minimum and navigates to the goal state. (**B**) One iteration of the array-based MCTS search. Note the state discretization results in branches snapping to the same points in space.

Next, we consider the wall-clock time performance of our array-based tree search on this MDP problem. In Fig. 5.7, we present the results of averaging the simulation times of the first 10 time steps of this experiment over 10 trials. We note that across a wide range of total simulations, our method demonstrates consistently better scaling with increased simulation depth than the tree-based method as well as the array-based method with no layer sorting. Interestingly, the array-based method with no sorting by layer has the same or worse scaling with maximum simulation depth than the tree-based implementation. Combined with its dramatically worse wall clock time, this suggest that sorting the nodes by layer is necessary for efficient memory access, and may indicate that the most frequently accessed nodes near the root of the search are in fact remaining within the processor cache.

Another general trend observed was that the tree-based search took longer during the later time steps of the experiment, while the array searches did not. This may be due to the tree more efficiently storing broader searches with more branching, which typically occurred at the beginning of the experiment, than the array-based implementations, which may be faster when the same values are repeatedly accessed in the CPU cache.

Finally, we compare the performance of our array-based method with tree-based implementations in partially observable planning settings. As a motivating problem, we implement FEAST using our array-based algorithm, and present the results in

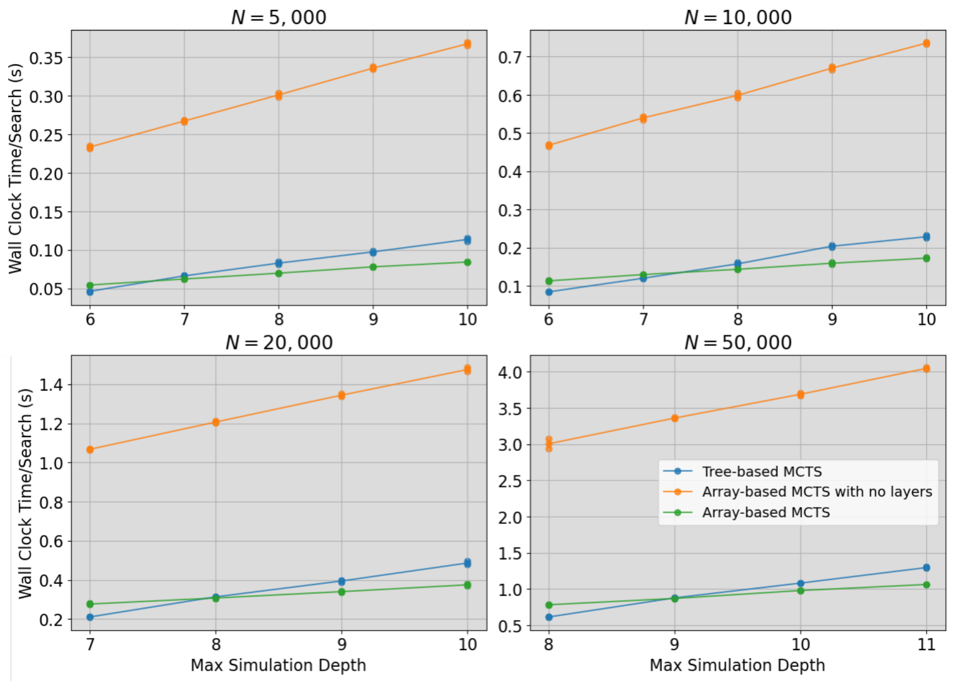


Figure 5.7: **Array-based MCTS performance on the non-convex problem.** The wall clock time of our array-based MCTS search compared with a tree-based implementation, and an alternative array-based MCTS algorithm that does not sort its nodes into layers. All times are averaged over 100 time steps (10 simulations of 10 steps each). Lower values indicate faster execution.

Fig. 5.8. Now we observe that a persistent performance gap remains between the two methods at all levels of simulation we considered in Fig. 3.3, and the improved scaling with simulation depth has disappeared. This may suggest that recomputing the belief at each node is prohibitively expensive or storing the large amount of data is eliminating the advantage our array-based method had. However, we also see that the performance gap is small, meaning that further optimizations, such as deployment on a GPU or improved algorithm design might lead to us outperforming the tree-based method.

5.5 Context with Related Work

There exists a significant body of work to accelerate MCTS and other tree search methods. One dominant area of research is methods of parallelizing tree search to take advantage of multi-threaded computing algorithms and processors such as GPUs, which excel at executing many operations in parallel. The challenge however, is that tree search methods are fundamentally sequential, as each search uses information gained during the previous backpropagation step. Multiple approaches have

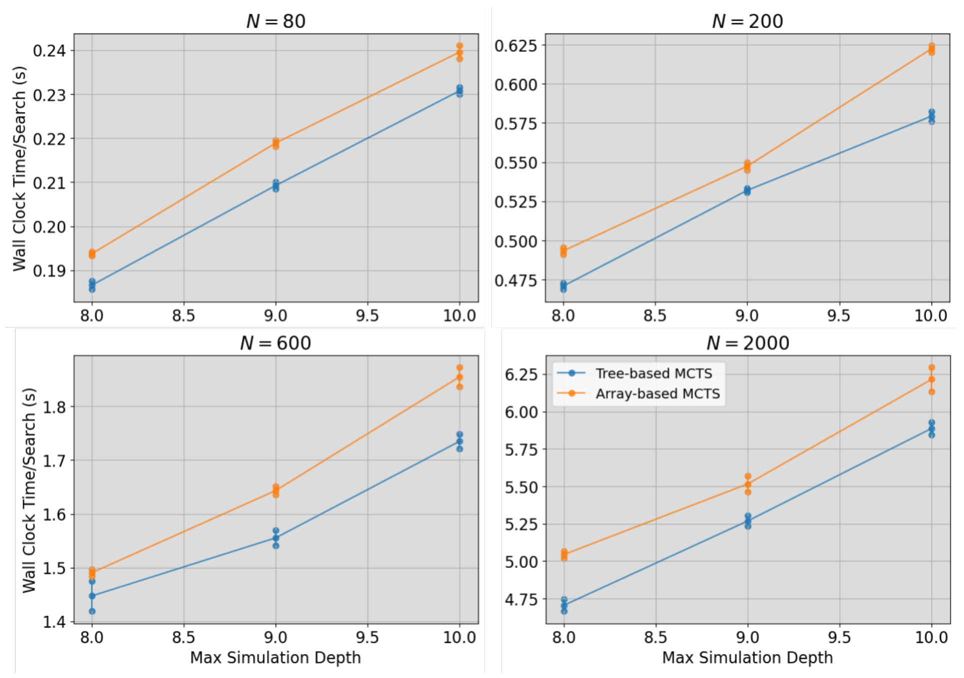


Figure 5.8: **Array-based MCTS performance on the active fault estimation problem.** The wall clock time of our array-based MCTS search compared with a tree-based implementation. All times are averaged over 100 time steps (10 simulations of 10 steps each). Lower values indicate faster execution.

been taken to parallelization. The simplest approach is root parallelization [123], where multiple independent tree searches are run. On completion, the value functions at each root node are averaged to determine the overall optimal value [124]. While conceptually simple, this approach is inefficient, as it results in many repeated computations. A more efficient approach is leaf parallelization [123], where roll outs from newly added leaf nodes are parallelized across multiple threads [124]. This minimizes repeat computation, but can only be employed at the end of the tree. Other approaches perform multiple searches within the same tree, but must carefully manage the communication between threads to avoid data corruption, introducing overhead [123]. Still other approaches combine several of these techniques, such as by using CPUs for in-tree parallelization, and GPUs for leaf parallelization [107], [115], [116].

Other approaches seek out heuristics to guide the tree search for more efficient exploration. Arguably the most well known instance of this approach is Alpha Go [125], which used learned value functions to approximate the outcome of a game from an intermediate position, giving MCTS earlier signals to bias its exploration towards. Similar approaches have been taken in real-time motion planning, where an expert tree search is used to train a neural network enabling more efficient online planning [126]. Still other approaches seek to leverage domain specific knowledge to modify the tree search, by limiting actions, setting sub goals, or imposing constraints [127]. While they achieve substantial empirical performance increases, parallelization, machine learning, and other heuristic methods loose the theoretical guarantees of traditional MCTS search [35] or more recent analysis [36]. This inspired our approach of optimizing the implementation of MCTS algorithms instead as a complementary approach. As we do not need to modify the search algorithm, we can run the original search algorithm and retain the theoretical guarantees or apply our implementation to these parallelization and heuristic methods as well.

Arrays have previously been proposed to compress tree searches in the context of string matching for natural language processing [128]. Branchless search algorithms have also previously been identified as desirable in the context of binary search problems [129], to avoid the penalties of branch misprediction on pipelined architectures [119]. However, these methods have yet to be applied to MCTS or decision making problems. Making branching predictable or avoiding it entirely is also a common technique in GPU optimization [130], [131], and we believe our array-based MCTS algorithm will enable better adaption of search methods to hardware accelerators. We will discuss this and other opportunities for future work in the next section.

5.6 Chapter Summary and Opportunities for Future Work

In this chapter, we have introduced an array-based implementation of MCTS. This method offers several advantages over traditional tree implementations. It requires no branch predictions, which means that it is suitable for processors with limited or poor branching capabilities such as GPUs. Its array-based data structure can also be efficient read and written too, and by storing each layer of the tree search separately, we increase the likelihood that the most frequently accessed nodes will remain in the cache.

In our experiments, we have validated that our method both successfully solves classic MDP problems, and out scales the tree-based implementation with increasing depth. In the POMDP setting, our algorithm achieved comparable speeds to the tree-based implementation, but a persistent gap of several tenths of a second remained regardless of the number of simulations or depth, which we suspected may have been due memory latency and the expensive recomputation of belief updates. This leads us to our first of several opportunities for future work.

While recomputation is inevitable if we wish to avoid branch prediction, one way to minimize the redundant effort is to identify which layers will be fully expanded by the end of the tree search. A hybrid approach could first expand these layers completely before computing the rest of the tree. On subsequent simulations, these first layers could be efficiently bypassed, and our predictable array-based search applied only to the layers which will not be fully explored. We anticipate a method like would enable our array-based search would outperform the tree-based implementation in the POMDP experiment as well.

While GPUs and other hardware accelerators with limited branching capabilities were a motivating application for the development of our method, the results presented here are all run on a CPU. Deploying our method on hardware accelerators is the most obvious opportunity for future work, and we anticipate significant performance improvements can be achieved. Similarly, dedicated profiling could be performed to identify other opportunities for algorithmic optimization, particularly to verify our intuition that keeping the portions of the tree close to the root in the processor cache is essential for rapid searches.

Chapter 6

EXTENSIONS TO ADVERSARIAL MULTI-AGENT SETTINGS

[1] J. Ragan, J. Ibrahim, S.-J. Chung, and F. Hadaegh, "Mitigating stealth attacks via game-theoretic switching in multi spacecraft systems," *International Astronautical Congress (Review at Acta Astronautica)*, 2024. [Online]. Available: `https://iafastro.directory/iac/paper/id/89213/summary/`,

[2] H. Tsukamoto, J. D. Ibrahim, J. Hajar, J. Ragan, S.-J. Chung, and F. Y. Hadaegh, "Robust optimal network topology switching for zero dynamics attacks," in *2024 63nd IEEE Conference on Decision and Control (CDC) (© 2024 IEEE)*, 2024. [Online]. Available: `https://arxiv.org/abs/2407.18440`,

In the pervious chapters, we developed FEAST and s-FEAST as methods for active fault estimation which enable robotic systems to autonomously test for anomalies by constructing trees of possible actions and observations, then selecting the branches which lead to the most informative (and safe) futures. While we developed this idea of planning active information gathering in single agent settings, our ideas naturally extend to multi-agent settings as well. One simple example would be to replace the comet obstacle in Chapter 3 with an independent agent, and modify the safety constraints to account for the actions (or range of actions) the other agent will take. While a more complicated problem, extensions like this are direct extension of the safe active fault estimation problem, and we expect s-FEAST to perform well in these settings as well, provided the computational power is scaled to match the problem complexity. In this chapter, we will instead explore applications of our active tree search methodology to a new class of inherently multi-agent problems.

6.1 Motivation

Formation flying spacecraft working collaboratively to achieve a mission objective present the possibility for greater performance than a single monolithic architecture [20], [21], in domains ranging from interferometry [132]–[134] to inspection [135]–[137]. At the same time, this increased system complexity presents increased opportunity for outside attacks.

Recent work has shown that spacecraft systems are vulnerable to cyberattacks [138], [139], including the possibility for full seizure of control [140]. For formation-flying missions this introduces the possibility of rouge agents acting against the group's objectives.

While formation flying can perform relative navigation to jointly estimate the state of the network [141], [142], each spacecraft cannot individually monitor every agent to ensure nominal performance. Instead, they must rely on collaborative sensing capabilities and information shared among the formation to establish an estimate of the full network's state. This reliance on trusted communication and collaboration presents an opportunity for a rouge actor to exploit, enabling it to avoid detection by sharing false observations.

Defending against stealth attacks is challenging, as a covert attack has both high knowledge about the system and the resources to exploit it [143]. Traditional approaches develop conditions under which no attack can be stealthy [144], [145], or which bound the effect of stealth attacks [146]–[149]. However in some systems, these conditions cannot be satisfied and the effects of stealth attacks may be unbounded. One such example is the angles only navigation problem we consider, where regardless of the selected observation topology, parts of the state space will be unobservable in the presence of false sensing data.

To address this, the system must be modified to reveal stealthy attacks. As changing the dynamics of the system is often infeasible, we instead consider switching the multi-agent network topology, and seek methods to find the optimal reconfiguration. In this chapter, we apply switching-based detection approaches to stochastic systems with nonlinear observations and a general probabilistic notion of stealthiness. As real spacecraft systems are inherently subject to noisy disturbances and measurements, the detection metric we propose provides a means for determining whether unexpected measurements are sufficiently anomalous to indicate the presence of an attack. An attacker trying to avoid detection may then try to "hide in the noise," leading to a two-player, zero-sum game where the attacker seeks to maximize the disruption without triggering detection, while the defender attempts to reveal the attack and bound the influence of any undetected attacks.

The remainder of the chapter is organized as follows. First, we develop a general framework for an observation switching system subject to control input attacks and false observations. We then show how switching between topologies can be formulated as a constrained optimization problem, much like the safe active fault estimation problem we considered in Chapter 3. When the attacker is also allowed to change strategies, we formulate the problem of stealth attacks in this system as a zero-sum game between the attacker and defender. Taking inspiration from s-FEAST and the methods we developed for safe active fault estimation, we then reformulate this problem into a form suitable for Monte Carlo Tree Search (MCTS) methods to develop the Switching System under Stealth Attacks Monte Carlo Tree Search (S3AM) algorithm which solves for both optimal attacks and defenses. Finally, we validate our algorithm by demonstrating the ability to limit the disruption of stealthy attacks in an angles only navigation system.

6.2 Problem Formulation

In the development of our method, we consider the following discrete-time system with linear dynamics and nonlinear sensing:

$$x_k = Ax_{k-1} + Bu_k + Ba_k + w_k \tag{6.1}$$

$$y_k^j = \begin{cases} h^j(x_k, d_k) + v_k & \text{if } j \neq \mathfrak{a} \\ y_k^a & \text{if } j = \mathfrak{a} \end{cases} \tag{6.2}$$

where the k subscript denotes a time index, A is the state transition matrix, B is the control influence matrix, h is the nonlinear observation function, $x \in \mathbb{R}^n$ is the state of the system, $u \in \mathbb{R}^m$ is the control input, $y \in \mathbb{R}^p \times \mathbb{R}^q$ are the observations indexed by j, and the stochastic process and measurement noise sequences w_k and v_k are assumed to be zero mean, mutually independent, and independently and identically distributed (i.i.d.).

This system is under attack, with malicious inputs, $a_k \in \mathcal{A}_k \subset \mathbb{R}^m$ and false observation data y_k^a replacing the $\mathfrak{a}$-th measurement. To detect these attacks, the defender can reconfigure the system, by selecting a new sensing topology $d_k \in \mathcal{D}$, changing the observation function h. The sets of possible attacks at each time step, $\mathcal{A}_k$, and topologies, $\mathcal{D}$, are assumed to be finite and the attacks are assumed to be bounded as $||a|| < a_{\max}, \forall a \in \mathcal{A}_k$.

The attacker and defender each carry a belief over the state of the system. Like in Chapter 2, we update these beliefs at each time step using a Bayesian update [31]:

$$b_k^d(x) = \frac{\mathbb{P}(y_k \mid x_k, d_k, u_k) \int \mathbb{P}(x_k \mid x_{k-1}, d_k, u_k) b_{k-1}^d(x) \mathrm{d}x_{k-1}}{\mathbb{P}(y_k \mid \overline{y}_{k-1}, \overline{d}_k, \overline{u}_k)} \tag{6.3}$$

$$b_k^a(x) = \frac{\mathbb{P}(\tilde{y}_k \mid x_k, d_k, u_k + a_k) \int \mathbb{P}(x_k \mid x_{k-1}, d_k, u_k + a_k) b_{k-1}^a(x) \mathrm{d}x_{k-1}}{\mathbb{P}(\tilde{y}_k \mid \overline{y}_{k-1}, \overline{d}_k, \overline{u}_k)} \tag{6.4}$$

The defender is unaware of the attack input, so updates its belief using u_k and y_k, whereas the attacker updates its belief using $u_k + a_k$, the true input to the system, and the unattacked (noisy) measurement $\tilde{y}_k$, which is defined as:

$$\tilde{y}_k^j = \begin{cases} y_k^j & \text{if } j \neq \mathfrak{a} \\ h^{\mathfrak{a}}(x_k, d_k) + v^{\mathfrak{a}} & \text{if } j = \mathfrak{a} \end{cases} \tag{6.5}$$

where $v_k^{\mathfrak{a}}$ indicates the (unknown) component of the process noise affecting the $\mathfrak{a}$-th measurement. The initial belief, $b_0(x)$, is shared between the attacker and defender, and is assumed to be unbiased. At each time step, the attacker is assumed to know the defender's belief, but not vice versa. For brevity, we will omit the explicit x dependence on beliefs going forward.

We define the stealthiness of the attacker by whether b_k^d given by Eq. (6.3) significantly differs from the nominal, unattacked prior belief:

$$b_k^n = \int \mathbb{P}(x_k \mid x_{k-1}, d_k, u_k) b_{k-1}^d(x) \mathrm{d}x_{k-1} \tag{6.6}$$

To determine statistical significance, we employ the Wasserstein-2 metric to compute the distance between these beliefs [150], [151]. We bound this distance as:

$$\mathbb{W}_2(b_k^d, b_k^n) \leq \alpha \tag{6.7}$$

where α is a user defined parameter set to limit the false alarm rate tolerable to the defender. Thresholding Wasserstein distance metrics as a method to detect stealth attacks has previously been proposed for linear time-invariant stochastic systems [152], [153]. Here we extend this approach to non-linear observations and switching systems.

In practice, evaluating the Bayesian updates in Eqs. (6.3) and (6.4) may be intractable, especially in nonlinear systems. Instead, we will consider nonlinear filters to propagate b^d and b^a. In this case, b^n corresponds to prior distribution of the defender's filter before measurement updates are.

Zero-Sum Game Formulation

This definition of attack stealthiness motivates us to understand how disruptive a stealth attack can be to the system given by Eqs. (6.1) and (6.2). To mitigate this risk, we seek well designed defensive strategies.

We formalize our problem as a two player game over K time steps. At each time step, the attacker seeks to maximize and the defender seeks to minimize a stage cost function, J_k, subject to this stealthiness constraint. The simple stage cost we consider is to regulate the system about a nominal position while minimizing the state disruption and the control effort of each agent:

$$J_k = (x_k - x_0)^\top Q_k (x_k - x_0) + u_k^\top R_k u_k \tag{6.8}$$

where x_0 is the nominal initial positions of each agent and Q_k and R_k are symmetric positive definite costs for the displacement and control effort. Summing these stage costs over the time horizon results in the following value function:

$$V(b_0, \pi_d, \pi_a) = \mathbb{E}\left[\sum_{k=1}^{K} J_k \mid b_0, \pi_d, \pi_a\right] \tag{6.9}$$

$$\begin{aligned}
&\text{s.t. Eqs. (6.1), (6.2), (6.3), (6.4),}\\
&\quad d_k \sim \pi_d(b_k^d), a_k \sim \pi_a(b_k^d, b_k^a)\\
&\quad u_k = g(b_k^d)\\
&\quad y_k^a = \mathbb{E}[h^{\mathfrak{a}}(x_k, d_k) \mid b_{k-1}^d, u_k, a_k = 0] + \hat{v}_k^{\mathfrak{a}}
\end{aligned} \tag{6.10}$$

where $V(b_0, \pi_d, \pi_a)$ represents the expected disruption of the attacker policy over K time steps, given the initial belief and defense policy. Note the difference with the value function defined in Chapter 2, as we now have two opposing players.

The control inputs are determined by the function g from the current defense belief. The false observation data y_k^a is designed to mimic the expected behavior of the attacked system, including additive noise $\hat{v}_k^{\mathfrak{a}}$ with the same distribution as the sensing noise v_k. The policies, $\pi_d \in \Pi_d$ and $\pi_a \in \Pi_a$, are stochastic maps from the information available to each player to sensing topologies and attacks, respectively. Importantly, these policies can incorporate reasoning about the opposing player's objectives, and the attacker is aware of the current defender belief b_k^d when planning a_k. As with the single player policies in the previous chapters, these policies are also closed-loop solutions which select new topologies and attacks at each time step in response to the updated beliefs stemming from the new observation.

We note that the attacker and defender consider opposite stage costs, making this a zero sum game. This is because we formulate the problem from the perspective of the defender maximizing its robustness to attack. In this case, the most adversarial attack is that designed to precisely disrupt the defender's objective. We summarize this and our other assumptions as follows:

Assumption 1. *We assume the following in formulating the zero-sum game.*

1. *Both the defender and the attacker know A, B, h, $\mathcal{D}$, $\mathcal{A}$, V and g, as well as the estimator and the distributions of w_k and v_k.*
2. *The attack a_k and false observations y_k^a are unknown to the defender and the topology d_k is unknown to the defender until after it selects a_k.*
3. *The belief of the defender b_k^d is know to the attacker when planning its attacks, creating its false observation and updating its belief, but the attacker's belief b_k^a is unknown to the defender.*
4. *Both the defender and the attacker know the parameters K, α, and $\mathfrak{a}$ as well as the reformulation parameters we will introduce in Section 6.3.*
5. *There are no other disturbances or uncertainties beyond the stochastic noise, attack input and false observation data in the system.*

These assumptions give both the attacker and defender full knowledge of the system dynamics and the attacker knowledge of the defender's current information, which enables stealth attacks and presents a worst-case scenario for the defender. Shared knowledge of the system parameters is a simplifying assumption, as it ensures the algorithm we develop to solve for both the best attack and defense is consistent. Assuming the defender is aware of which agent will be compromised, if any, is a reasonable assumption when grounds for suspicion exist, such as previous anomalous behavior or a documented vulnerability. However, lifting this assumption and generalizing to suspecting all agents simultaneously is an opportunity for future work.

Considering Secondary Objectives

In addition to minimizing or maximizing the disruption to the system that is captured by value function of Eq. (6.9), the defender or the attacker may have additional objectives, such as preventing or maintaining stealth, controllability, observability, or more. One approach to achieving these objectives is to specify individual metrics for each goal, such as the following observability criterion:

$$J_{k,\text{obs}} = y_k^T y_k \tag{6.11}$$

which quantifies how large the observation signal is at timestep k. Since a clearly observable signal may be desirable to the defender, we could modify the stage cost (Eq. (6.8)), to instead be:

$$J_k = (x_k - x_0)^\top Q_k (x_k - x_0) + u_k^\top R_k u_k - cJ_{k,\text{obs}} \tag{6.12}$$

and the value function in Eq. (6.9) accordingly to represent that minimizing disruption and control effort is still good, but now we also want large observability. But this leads to a challenge. How should we appropriately quantify c to properly weigh the the competing objectives? Further, since observability is desirable, we now have a negative term in the stage cost, so Eq. (6.12) is not convex. Worse, the cost is potentially unbounded, as depending on the observation function h, infinite $J_{k,\text{obs}}$ may be possible with finite disruption and control effort, or at least scale more rapidly regardless of choice of c.

The challenge illustrated by this example is a common issue with multi-objective optimization. From the intuition we developed in Chapter 3, an effective solution is to consider transforming the secondary objectives into constraints instead. Like with the safety thresholds we considered in the safe active fault estimation problem, stealth, controllability and observability, are all better formulated as constraints. In the example above, while observability is desirable, beyond a minimum level we do not need a larger signal.

We can apply this intuition to our zero-sum game to impose on the attacker a stealth constraint (Eq. (6.7)) that it must satisfy to avoid losing by default, and formulate the two player game in the minimax sense. The defender seeks topologies that minimize the disruption that an optimal attacker can cause, and can win by detecting the attack.

This formulation provides a robust defense policy similar to the approach of $\mathcal{H}_\infty$ control to minimize the effect of the worst-case disturbance [154].

Definition 10 (Robust stealth attack zero-sum game)**.** *The problem of robustly selecting defense policies to minimize the disruption of an adversarial attacker is formalized as:*

$$\min_{\pi_d \in \Pi_d} \max_{\pi_a \in \Pi_a} \mathbb{E}\Bigg[\prod_{k=1}^{K} \Big(\mathbb{1}_\alpha(b_k^d, b_k^n) \Big) V(b_0, \pi_d, \pi_a) \mid b_0, \pi_d, \pi_a \Bigg] \tag{6.13}$$

where $\prod$ indicates a product, and the the indicator over α-stealthy attacks $\mathbb{1}_\alpha(b_i^d, b_i^n) = 1$ if $\mathbb{W}_2(b_i^d, b_i^n) \leq \alpha$ and 0 otherwise.

Note that by construction, $\prod_{k=1}^{K} \Big(\mathbb{1}_\alpha(b_k^d, b_k^n) \Big) = 1$ only if every time step up to and including time K satisfies the stealthiness constraint. Also, from the definition of Eq. (6.9) the value function in Eq. (6.13) can be factored out of the expectation. Since the number of defense topologies and attacks is finite, the policy sets are finite-dimensional compact sets. The cost functional is continuous (and jointly convex) in each of its arguments. Therefore, by Theorem 4.7 of [155], this game satisfies the sufficient conditions for the existence of at least one mixed Nash equilibrium. Further, the minimax formulation of Eq. (6.13) can equivalently be posed as a maximin problem, considering the best attack to respond to a chosen defense [156].

6.3 Problem Reformulation

While Definition 10 formalizes our problem, solving the robust stealth attack zero-sum game may be numerically intractable. This motivates us to reformulate the problem and develop a MCTS method to asymptotically approximate optimal solutions, as we did in Chapters 2 and 3. In addition to the single player cases we considered previously, MCTS methods have a rich history when applied to two player games [34], [127], [157], and can be shown to converge to the optimal minimax value [34], [158] in two player, zero-sum games. These two player formulation maintain their desirable asymptotic convergence properties and still only require access to a system model and a stage cost function to search for optimal solutions [35], making them well suited to solving the robust stealth attack zero-sum game.

However, the MCTS methods we employ require value functions that can be decomposed into a stage cost and a cost to go, as $V(b_0) = J(b_0) + \mathbb{E}[V(b_1)]$. Equation (6.13) cannot be, as $\prod_{k=1}^{K} \Big(\mathbb{1}_\alpha(b_k^d, b_k^n) \Big)$ is a function of all time steps up to the horizon. Further, the MCTS analysis we adopt assumes stage costs bounded between 0 and 1. This motivates the problem reformulations we present in this section.

Equivalent Reformulation

To make the problem posed by Eq. (6.13) compatible with these solvers, we adopt the following transformation of the stage cost (motivated by the reformulation we made in Chapter 3 when developing s-FEAST):

$$J_k^\alpha = \prod_{\kappa=1}^{k} \left(\mathbb{1}_\alpha (b_\kappa^d, b_\kappa^n) \right) \left(J^s + (1 - J^s) J_k^{\text{sat}} \right) \tag{6.14}$$

$$J_k^{\text{sat}} = \frac{J_k}{1 + J_k} \tag{6.15}$$

where $J^s = \frac{K}{K+1}$, and J_k^{sat} is a saturated stage cost that is a strictly monotonic transformation of J to the interval $[0, 1)$. We define $V_\alpha(b_0, \pi_d, \pi_a)$ as the value function given by Eq. (6.9), with J replaced by J^α.

Given this transformation, we have the following theorem:

Theorem 6 (Equivalent unbounded reformulation)**.** *The Nash equilibria of the following problem with the transformed stage cost given by Eq.* (6.14)*, are also Nash equilibria of the robust stealth attack zero-sum game:*

$$\min_{\pi_d \in \Pi_d} \max_{\pi_a \in \Pi_a} V_\alpha(b_0, \pi_d, \pi_a) \tag{6.16}$$

Proof. The saturation function given by Eq. (6.15) maps the original cost J_k, which is non-negative, to the interval $[0, 1)$. This transformation is strictly increasing and continuous for all $J_k \geq 0$. As $J_k \to 0, J_k^{\text{sat}} \to 0$, and as $J_k \to \infty, J_k^{\text{sat}} \to 1$. Therefore, the saturation function is a strictly increasing and continuous transformation, preserving the order of best responses.

The full transformation is given by Eq. (6.14). When the stealth constraint is satisfied, it is an affine transformation of the saturated cost J_k^{sat} with positive coefficients. Since affine transformations preserve monotonicity and the saturation function (Eq. (6.15)) is strictly increasing, Eq. (6.14) is also strictly increasing in J_k.

Under strictly monotonic and affine transformations, Nash equilibria are invariant [159]–[161]. Thus, the transformed and untransformed games share the same Nash equilibria.

When the stealth constraint is not satisfied, the transformation is not strict. However, equilibria not satisfying the stealth constraint represent scenarios where the defender can win by revealing the attack. By construction of J^s, the minimum value of any combination of polices which are expected to remain stealthy is KJ^s, which is higher than the maximum expected value of any policy combination where the attack is expected to be detected, $K-1$, as $KJ^s = \frac{K^2}{K+1} > \frac{K^2-1}{K+1} = K-1$. Therefore, under this transformation equilibria which are stealthy remain strictly separated from those which are not. □

This equivalence means methods that converge to the solutions of the transformed problem, such as the MCTS algorithm we present in Section 6.4 also solve the original robust stealth attack zero-sum game.

Chance Constraint Modification

To a stealthy attacker, Eq. (6.7) represents a constraint that it must satisfy, since the above reformulation results in stealthy attacks always outperforming those that are detected. However, in the presence of noise and stochastic polices like we consider in this chapter, an attacker cannot guarantee satisfaction when planning in advance. This can lead to a trivial solution where the attacker prioritizes stealth above disruption, taking no actions. Therefore, we modify the robust stealth attack zero-sum game given by Definition 10 so that the attacker seeks to satisfy the following chance constraint on the probability of detection:

$$\mathbb{P}(\mathbb{W}_2(b^d_k, b^n_k) > \alpha \mid b_0, \pi_d, \pi_a) < \beta, \forall k \tag{6.17}$$

where $\beta \in (0, 0.5)$ is an attacker specified risk. Note that to avoid the trivial solution of no attacks, the risk should be higher than the false alarm rate of the unattacked system, or this condition cannot be satisfied. The modified problem is:

Definition 11 (Chance-constrained robust stealth attack zero-sum game)**.** *The chance-constrained robust stealth attack zero-sum game is given by:*

$$\min_{\pi_d \in \Pi_d} \max_{\pi_a \in \Pi_a} \mathbb{E}\left[\prod_{k=1}^{K} \left(\mathbb{1}^{\beta}_{\alpha}(b^d_k, b^n_k) \right) V(b_0, \pi_d, \pi_a) \mid b_0, \pi_d, \pi_a \right] \tag{6.18}$$

where $\mathbb{1}^{\beta}_{\alpha}(b^d_k, b^n_k) = 1$ *if* $\mathbb{P}(\mathbb{W}_2(b^d_i, b^n_i) > \alpha) < \beta$ *and 0 otherwise.*

This modified problem can also be transformed to an equivalent problem compatible with MCTS in the same manner as Theorem 6.

Corollary 1. *The Nash equilibria of the following problem are also Nash equilibria of the chance-constrained robust stealth attack zero-sum game:*

$$\min_{\pi_d \in \Pi_d} \max_{\pi_a \in \Pi_a} V_\alpha^\beta(b_0, \pi_d, \pi_a) \tag{6.19}$$

where V_α^β is defined by replacing $\mathbb{1}_\alpha$ in V_α with $\mathbb{1}_\alpha^\beta$.

Proof. The only difference between the robust stealth attack zero-sum game (Definition 10) and the chance-constrained robust stealth attack zero-sum game (Definition 11) is the definition of the indicator functions, $\mathbb{1}_\alpha$ and $\mathbb{1}_\alpha^\beta$. Since Theorem 6 does not depend on the choice of indicator function, the reformulation holds for both problems. □

This modification allows our MCTS based method to plan defenses against attackers that risk detection, which will be more disruptive to the system. However, to analyze the effectiveness of our defense, we will use the original transformed metric give by Eq. (6.14). The risk tolerance of the attacker is not important to from the perspective of the defender, instead we wish to quantify how well we minimize the effect of stealthy attacks.

Conservative Stealth Approximation

While the previous reformulations in Sections 6.3 and 6.3 make the robust stealth attack zero-sum game compatible with MCTS solvers, evaluating $\mathbb{1}_\alpha^\beta$ may be intractable when planning within a tree. Instead, we adapt the theory in Chapter 3 to develop a conservative chance constraint on the probability of detection using a finite sample approximation. For each new attack considered, M simulations are performed from the perspective of the attacker. For each i-th simulation, a possible state x_{k-1} is drawn from b_{k-1}^a, and propagated through the system model (Eqs. (6.1) and (6.2)) along with u_k and the (assumed) d_k to generate an observation and update b_k^d according to Eq. (6.3). We then compute $W_i = \mathbb{W}(b_k^d, b_k^n)$ for each sample and define the sample mean and standard deviation of the Wasserstein-2 distance as:

$$\hat{\mu}_W = \frac{1}{M}\sum_i W_i \tag{6.20}$$

$$\hat{\sigma}_W^2 = \frac{M+1}{M(M-1)}\sum_i (W_i - \hat{\mu}_W)^2 \tag{6.21}$$

We claim that $\hat{\mu}_W$ is an unbiased estimator of $\mathbb{W}_2(b_k^d, b_k^n)$, so when $\hat{\mu}_W > \alpha$, the sample mean indicates an attack is present with $> 50\%$ probability. Since $\beta \in (0, .5)$, we focus our attention on the case where $\hat{\mu}_W < \alpha$.

Theorem 7 (Conservative finite sample approximation of stealth)**.** *Adapting the finite sample approximation of Chebyshev's Inequality first developed by Saw* et al. *[46] and simplified by Kabán [47], we can conservatively achieve this bound with the following condition when $\hat{\mu}_W \leq \alpha - \hat{\sigma}_W$ and $M \geq 2$:*

$$\frac{1}{M+1}\left\lfloor \frac{M+1}{M}\left(\frac{\hat{\sigma}_W^2(M-1)}{(\alpha - \hat{\mu}_W)^2} + 1\right)\right\rfloor \leq 1 - \beta. \tag{6.22}$$

Proof. Considering the distance metric between the nominal and realized beliefs as a random variable W resulting from the system noise and the possibly stochastic policies, the condition for remaining stealthy is $\mathbb{P}(W > \alpha) < \beta$. Under the assumption $\hat{\mu}_W < \alpha$, we subtract the empirical mean and upper bound the one sided tail probability with a two-sided condition:

$$\begin{aligned}\mathbb{P}(W > \alpha) &= \mathbb{P}(W - \hat{\mu}_W > \alpha - \hat{\mu}_W) \\ &\leq \mathbb{P}(|W - \hat{\mu}_W| > \alpha - \hat{\mu}_W) \end{aligned} \tag{6.23}$$

Using Eq. (3.17), and choosing $\lambda = \frac{\alpha - \hat{\mu}_W}{\hat{\sigma}_W}$ we have:

$$\begin{aligned}&\mathbb{P}(|W - \hat{\mu}_W| > \alpha - \hat{\mu}_W) \\ &\quad \leq \frac{1}{M+1}\left\lfloor \frac{M+1}{M}\left(\frac{\hat{\sigma}_W^2(M-1)}{(\alpha - \hat{\mu}_W)^2} + 1\right)\right\rfloor \end{aligned} \tag{6.24}$$

Combining Eqs. (6.23) and (6.24) completes the proof. □

In addition to the shared parameters listed in Assumption 1, the attacker and defender are assumed to also know the parameters β and M and are aware of the transformed value function V_α^β.

6.4 Main Method (S3AM Algorithm)

In this section, we present the Switching System under Stealth Attacks Monte Carlo Tree Search (S3AM) Algorithm and the adaptions made from existing partially observable tree search algorithms to solve zero-sum stealth constrained games. Given the corresponding information, our algorithm converges to both the defense and attack policies in Eq. (6.9).

S3AM Algorithm

We start with the algorithmic structure, full Bayesian updates, and chance constraints of s-FEAST (Chapter 3), and adapt it to the two player adversarial setting, as presented in Algorithm 36 below. This pseudocode solves for both the optimal defense and attack. Note that while the tree first selects a defense and then an attack at each time step, we could equivalently select an attack first and then a defense. This is because neither player is aware of the other's selection until they are simultaneously applied to the system, and the cost function in Definition 10 and its reformulations are continuous and convex, [156].

In the algorithm's notation, a node is defined as an ordered history H of defense topologies, attacks, and observations, with corresponding number of visits $N(H)$ and value estimate $\hat{V}(H)$. Each simulation is performed from the root node until the depth, δ, reaches the maximum depth K. Defenses and attacks are selected according to the UCT augmented value function with exploration parameter c [35]. New states and observations are simulated by the model-based generator G which encodes Eqs. (6.1),(6.2),(6.10) and discretizes the observations to yield finite observation branching, which is necessary to avoid degenerate tree searches [162]. Both the defender and attacker beliefs b^d and b^a are updated throughout the tree, with the attacker belief treated as the system ground truth, while the defender belief is used to determine if detection has occurred. When the defender performs the tree search, b^a is assumed to be the same as b^d at initialization. When the attacker performs the tree search, it can provide its knowledge of the true b^a. The nominal belief b^n is predicted by the filter prior function $\mathcal{F}^-[d_{\delta+1}, u_{\delta+1}, b^d]$ at each time step given the defense, input and previous belief. $\mathcal{F}[d_{\delta+1}, y_{\delta+1}, u_{\delta+1}, b^d]$ represents the filter updates given measurement $y_{\delta+1}$.

When a previously unexplored history is encountered, the simulation rolls out to the max depth by uniformly sampling random actions from the defense topology and attack sets $\mathcal{D}$ and $\mathcal{A}$. For each new attack considered, M simulations are

Algorithm 8: S3AM

```
globals: $\hat{V}(\cdot) \leftarrow 0,\ N(\cdot) \leftarrow 0,\ S(\cdot) \leftarrow \emptyset$
def search($\mathfrak{a}, b^d, b^a = \emptyset$):
    if $b^a = \emptyset$ then $b^a = b^d$;
    for $i \leftarrow 1$ to $N$ do
        simulate($\emptyset, 0, \mathfrak{a}, b^d, b^a$) ;
    $d^* = \arg\min_d \hat{V}(\{d\})$ ;
    $a^* = \arg\max_a \hat{V}(\{d^*, a\})$ ;
    return $d^*, a^*$ ;
def simulate($H_\delta, \delta, \mathfrak{a}, b^d, b^a$):
    if $\delta > K$ then return 0 ;
    $x_\delta \sim b^a$ ;
    $d_{\delta+1} \leftarrow \arg\max_d -\hat{V}(H_\delta \cup d) + c\sqrt{\frac{\log N(H_\delta)}{N(H_\delta \cup d)}}$;
    $a_{\delta+1} \leftarrow \arg\max_a \hat{V}(H_\delta \cup \{d_{\delta+1}, a\}) + c\sqrt{\frac{\log N(H_\delta \cup d_{\delta+1})}{N(H_\delta \cup \{d_{\delta+1}, a\})}}$;
    $u_{\delta+1} \leftarrow g(b^d)$;
    if $S(H_\delta \cup \{d_{\delta+1}, a_{\delta+1}\}) = \emptyset$ then
        $b^n = \mathcal{F}^-[d_{\delta+1}, u_{\delta+1}, b^d]$;
        for $i \leftarrow 1$ to $M$ do
            $(\cdot, \cdot, b^d, \cdot)$ = propogate($x_\delta, u_{\delta+1}, a_{\delta+1}, d_{\delta+1}, \mathfrak{a}, b^d, b^a$) ;
            $W_i \leftarrow \mathbb{W}(b^d, b^n)$;
        $\hat{\mu}_W, \hat{\sigma}_W \leftarrow$ sampleStatistics($\{W_1, ..., W_M\}$) ;
        $S(H_\delta \cup \{d_{\delta+1}, a_{\delta+1}\}) \leftarrow$ approxStealthCond($\hat{\mu}_W, \hat{\sigma}_W, M, \alpha, \beta$) ;
    $x_{\delta+1}, y_{\delta+1}, b^d, b^a$ = propogate($x_\delta, u_{\delta+1}, a_{\delta+1}, d_{\delta+1}, \mathfrak{a}, b^d, b^a$) ;
    $H_{\delta+1} \leftarrow H_\delta \cup \{d_{\delta+1}, a_{\delta+1}, y_{\delta+1}\}$ ;
    $\hat{V}_{\delta+1} \leftarrow J^s + (1 - J^s) * J^{\text{sat}}(x_{\delta+1}, u_{\delta+1})$;
    $\hat{V}_{\delta+1} \leftarrow \hat{V}_{\delta+1} * S(H_\delta \cup \{d_{\delta+1}, a_{\delta+1}\})$ + simulate($H_{\delta+1}, \delta + 1, \mathfrak{a}, b^d, b^a$);
    $N(H_\delta) + +$ ;
    $N(H_\delta \cup d_{\delta+1}) + +$ ;
    $N(H_\delta \cup \{d_{\delta+1}, a_{\delta+1}\}) + +$ ;
    $\hat{V}(H_\delta \cup d_{\delta+1}) \leftarrow \hat{V}(H_\delta \cup d_{\delta+1}) + \frac{\hat{V}_{\delta+1} - \hat{V}(H_\delta \cup d_{\delta+1})}{N(H_\delta \cup d_{\delta+1})}$ ;
    $\hat{V}(H_\delta \cup \{d_{\delta+1}, a_{\delta+1}\}) \leftarrow \hat{V}(H_\delta \cup \{d_{\delta+1}, a_{\delta+1}\}) + \frac{\hat{V}_{\delta+1} - \hat{V}(H_\delta \cup \{d_{\delta+1}, a_{\delta+1}\})}{N(H_\delta \cup \{d_{\delta+1}, a_{\delta+1}\})}$ ;
    return $\hat{V}_{\delta+1}$ ;
def propogate($x_\delta, u_{\delta+1}, a_{\delta+1}, d_{\delta+1}, \mathfrak{a}, b^d, b^a$):
    $(x_{\delta+1}, y_{\delta+1}) \sim G(x_\delta, u_{\delta+1}, a_{\delta+1}, d_{\delta+1})$ ;
    $b^a \leftarrow \mathcal{F}[d_{\delta+1}, y_{\delta+1}, u_{\delta+1} + a_{k+1}, b^a]$ ;
    $y^{\mathfrak{a}}_{\delta+1} \leftarrow \mathbb{E}[h^{\mathfrak{a}}(x_k, d_k) \mid b^d, u_{\delta+1}, a_{\delta+1} = 0] + \hat{v}^{\mathfrak{a}}_k$ ;
    $b^d \leftarrow \mathcal{F}[d_{\delta+1}, y_{\delta+1}, u_{\delta+1}, b^d]$ ;
    return $x_{\delta+1}, y_{\delta+1}, b^d, b^a$ ;
```

performed to compute the sample statistics via sampleStatistics, which follows Eqs. (6.20),(6.21). These statistics are used to empirically estimate the stealth chance constraint (Eq. (6.17)) using approxStealthcond which returns the boolean output of Eq. (6.22). As with FEAST and s-FEAST, to save recomputing the belief propagation at each node, our algorithm saves the nodes encountered during roll outs instead of discarding them. After completing all N simulations (the level of planning), the defense which minimizes the maximum disruption from an attack and that corresponding attack are returned. The defender and attacker each apply their respective solutions to the system. The resulting observation is used to update the system's beliefs, and a new tree is planned from this new root node to select the next defense and attack. A diagram of our tree search is provided below in Fig. 6.1.

Motivating Problem: Angles Only Relative Navigation

In this section, we develop the angles only relative navigation problem we consider in our simulations. A group of P spacecraft with two-dimensional double integrator dynamics are collaboratively estimating the joint state of the system, and seeking to minimize the disruption of the formation. However, one of the agents is compromised by the attacker. For this adversary agent, the attacker can select attack inputs to disrupt the formation and will also report false observations to hide its activities. For this system, the state and control influence matrices in Eq. (6.1) are given as:

$$A = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_P \end{bmatrix}, \; B = \begin{bmatrix} B_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & B_P \end{bmatrix} \tag{6.25}$$

$$A_j = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; B_j = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ \Delta t & 0 \\ 0 & \frac{\Delta t^2}{2} \\ 0 & \Delta t \end{bmatrix} \tag{6.26}$$

where Δt is the time step between time $k-1$ and k, A_j is the block dynamics matrix and B_j the block control influence matrix for the j-th spacecraft. The corresponding states $x \in \mathbb{R}^{4P}$, control action $u \in \mathbb{R}^{2P}$, and observation $y \in \mathbb{R}^{2P}$ are similarly defined per agent, with x_k^j, u_k^j, and y_k^j indicating the component corresponding to the j-th agent. Let $\mathfrak{a}$ again be the index of the attacker. The control and observations are:

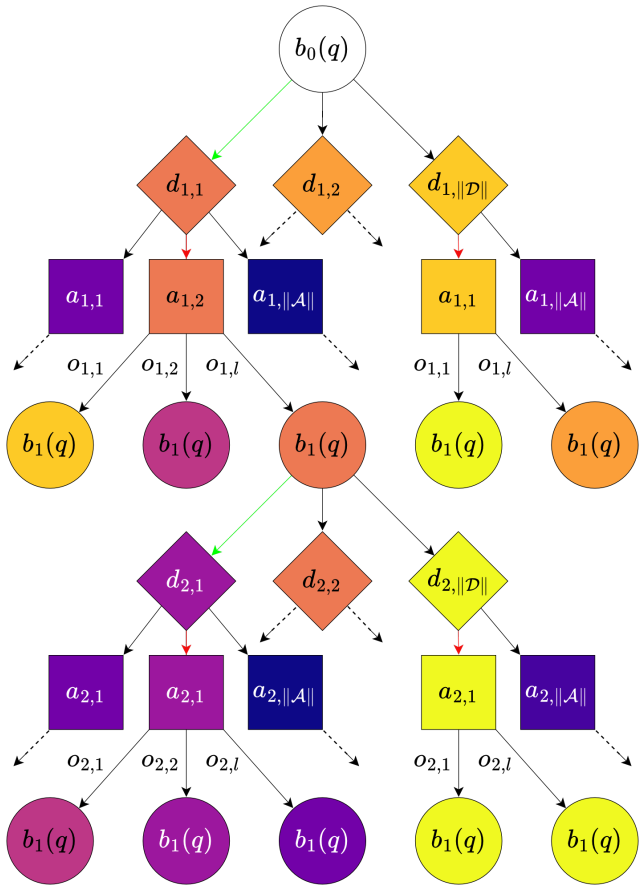


Figure 6.1: **Diagram of the S3AM algorithm.** Here, brighter nodes represent system states which are more disruptive and stealthy, while darker nodes are less disruptive or less stealthy. Circular nodes represent belief nodes, from which a defense node (diamond shaped) is selected by the defender from the defense set. The attacker then selects an attack node (square shaped) from the attack set. A defense node has the same color as the worst (brightest) attack under it. The attack has the same color as the average belief node under it. When there is no exploration, the defender select the darkest node available, represented by the green arrows, aware that the attacker will select the brightest node available, represented by the red arrows. Exchanging the order of defense and attack selection results in an equivalent tree search [156].

$$u_k^j = -K_{\text{LQR}}\mathbb{E}[x^j \mid b_k^d] \tag{6.27}$$

$$y_k^j = \begin{cases} [\angle_{j,d_k^j}, 0] + v_k^j & \text{if} j \neq \mathfrak{a}, d_k^j \neq P+1 \\ [\angle_{j,d_k^j}, \|x_k^j\|] + v_k^j & \text{if } j \neq \mathfrak{a}, d_k^j = P+1 \\ y_k^a & \text{if } j = \mathfrak{a} \end{cases} \tag{6.28}$$

where K_{LQR} is the LQR gain resulting from the Q_k and R_k terms of the cost function and $\mathbb{E}[x^j \mid b_k^d]$ is the defender's estimate of the j-th agent's state. For this system, the sensing topology determines which agent each spacecraft is currently looking at, with $d_k \in [1, \ldots, P+1]^P$ and $d_k^j \neq j$ the target of the j-th spacecraft. With this, $\angle_{j,d_k^j}$ is the angle between each spacecraft and its target. The origin of the relative navigation is the $P+1$ reference target, and observing this reference also provides a line of sight distance. When looking elsewhere, the spacecraft is aware the range measurement is 0, representing no information. This is necessary to provide a scale for the formation, and the reference target could represent a central trusted spacecraft, an inspection target, or some other important feature. The attack set at each time step is given by:

$$\mathcal{A}_k = \mathcal{A} + K_{\text{LQR}}\mathbb{E}[x^{\alpha} \mid b_k^d] \tag{6.29}$$

where $\mathcal{A}$ is the set of common attack actions at every time step. In effect, the adverse agent ignores the nominal control action and selects its own independent attack action.

6.5 Results

In this section, we study the chance-constrained robust stealth attack zero-sum game in our angles only relative navigation system and demonstrate the performance of our proposed algorithm through numerical simulations.

Overview of Experiments

We consider three different scenarios. First, to establish a false alarm rate, we consider the distributed angles only navigation system when there are no attacks present. Using this experiment to determine a threshold for detection, we then consider how planning allows the attacker to disrupt the system and avoid detection when employed against a baseline approach of randomly selecting a defense topology. In our final scenario, we show that by employing our proposed method to plan defenses over a horizon, the defender can successfully limit the effect of stealthy attacks on our system, by forcing the attacker to cause less disruption or be revealed. We first analyze the quantitative results of each experiment then provide qualitative examples a the end of the section.

The defenses are selected from a set of eight topologies, designed so that each agent inspects and is inspected by its neighbors equally often, and so that every agent looks at the reference target in two different topologies. The attacks are selected from an attack set of five options: either to take no action, or apply an acceleration of ±0.01 m/s^2 in both the x and y directions. The magnitude of this attack input is scaled to match the process and sensor noise, which are zero-mean Gaussians. As a baseline, we also consider the effectiveness of randomly selected defenses and attacks from these sets. For each experiment, the attacker's tolerable detection threshold is $\beta = 0.25$). Further implementation details are provided in Appendix D.

Unattacked System

To determine an appropriate value of α, we consider our first scenario, where no attacker is present in the system. The results are shown in Fig. 6.2 below. In Fig. 6.2A, we see the disruption in the system grows initially due to the process and sensor noise, before reaching a steady state value as a result of the feedback control. In Fig. 6.2B, we see the same occurs for the Wasserstein-2 distance, where the steady state value of 0.12 with a standard deviation of 0.09 is reached after 17 time steps. We use this to set the detection threshold to a value of $\alpha = 0.5$, which is more than 3 standard deviations from this sample mean. Given this threshold, the false alarm rate is shown in Fig. 6.2C, which shows the fraction of trials which have had no detection events at any time up to and including the current time step. Once steady state conditions are reached, a steady false alarm rate of 0.66% per second occurs. Because outlier noise values are possible, though unlikely, false alarms are to be expected and the value of α must be tuned by the defender to balance false positives and sensitivity to attacks. We envision our method being used to plan in the near-term, when we have reason to suspect a malicious agent may be present. Therefore, we target a 5% false positive rate over a 20 time-step duration, and see we achieve a rate of 5.2% as well as converged behavior in these experiments. We will use this horizon and $\alpha = 0.5$ for the remaining experiments.

Effect of Planning on the Attacker

To demonstrate the difficulty of detecting an attacker which plans stealthy attacks, we consider the baseline of random defense selection against attackers of increasing levels of planning in Fig. 6.3 below. In this scenario, we first consider an attacker which randomly selects from the attack set. We see in Fig. 6.3A this leads to rapid disruption of the formation. This is unsurprising, as a random attack input leads to random walk-like behavior, which is a stochastic process well known to exhibit unbounded variance [163]. However, Fig. 6.3B&C shows this comes at the price of

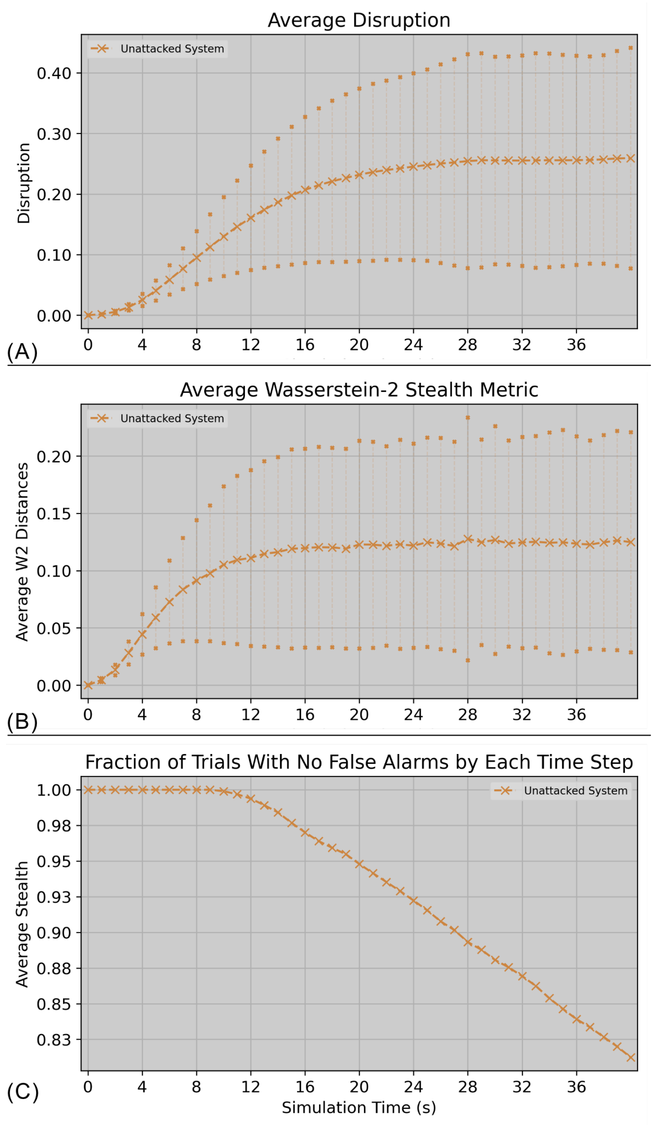


Figure 6.2: **Unattacked system results.** The disruption, Wasserstein-2 stealth metric and false alarm rate are averaged over 6000 simulations, and the $1-\sigma$ error region is shown.

dramatically reduced stealthiness. In Fig. 6.3D, we see this lack of stealthiness also results in poor performance on the transformed disruption metric given by Eq. (6.14), demonstrating that a random attacker cannot efficiently solve the chance-constrained robust stealth attack zero-sum game.

Instead, we employ our tree search method given by Algorithm 36, which conservatively plans the most disruptive attack given the best possible defense the random defender might employ as well as the anticipated future results. We see in Fig. 6.3A that the overall disruption that results form this strategy is decreased compared to a random attacker that does not consider stealthiness, but still grows steadily while minimizing detection (Fig. 6.3B&C) and achieving a high transformed disruption (Fig. 6.3D).

We note in Fig. 6.3C that for all levels of planning, the attacker satisfies the desired point-wise stealthiness of 25%. However, alarms are raised more frequently than the unattacked system, at least 3.28 times more often by the end of the experiment (Fig. 6.3B). This matches our intuition that the attacker must take risks to disrupt the system, and validates the detection threshold selected, as the increased alarm rate even in the presences of stealthy attacks indicates the detection metric is succeeding at identifying anomalous behavior. We also see a direct correlation between point-wise stealth shown in Fig. 6.3C and overall stealth in Fig. 6.3C, suggesting attackers that are detected once are more likely to be detected again. This validates the structure of our transformed disruption metric and tree search, which labels all future states as not stealthy once an alarm has been raised.

Noticeably, increasing the planning level of the attacker does not improve the observed performance on any metric. In fact, higher values of planning of $N = 5000$ and $10,000$ actually slightly worsen the observed performance. This is possibly due to the higher levels of planing being overly conservative when considering the best defense possible. It may also be due to the solver encountering a local minima by prematurely ruling out possible attacks. In the next scenario we will see similar local minima behavior with increased resources for the defender until very high levels of planning are achieved. Because of this trend, we select $N = 2000$ as the level of adversary planning to compare our planned defenses against.

Effect of Planning on the Defender

In the last simulation we consider, we show the effect of increased planning on the ability to select defenses to best limit the disruption of a stealthy attacker. In Fig. 6.4 below, we see that all levels of planning initially out perform the random defender in both reducing stealthiness of the attacker and the transformed disruption metric, with the highest level of planning, $N = 20,000$, maintaining the performance gap throughout the experiment. However, increasing the level of planning does not uniformly increase the performance of the defender. Like with the attack planner, higher levels of planning initially lead to worse performance, in this case in the form of increased stealthiness for the attacker. It is not until the highest level of planning, $N = 20,000$, that consistently lower levels of attack stealthiness are achieved throughout the experiment (Fig. 6.4B), along with lower levels of the transformed disruption metric (Fig. 6.4C). This indicates there may again be a local minima present in the tree search that high levels of planning are needed to escape, or that an alternative strategy is being discovered. Together, these results validate that planning outperforms the random baseline and that sufficiently high levels of planning yield further improvements. It also establishes that random defenses are a strong baseline, and indicates the tree quickly rediscovers this behavior. This is not unsurprising, as randomness has been employed in real-world settings to increase the effectiveness of security screenings [164], [165], and efficiently use existing resources [166].

Due to computation limits, particularly the memory requirements of the larger tree searches, increased levels of planning were not explored. Optimizing the implementation of the S3AM was not an objective of this work, but our investigation suggests it is a promising opportunity for future work.

Qualitative Experiments

In this section we look at several qualitative experiments to better understand the behavior that leads to successful stealthy attacks and detections in this system. In Fig. 6.5 below, we see a simulation between a random attacker and defender. At the shown time, the agent controlled by the random attacker has drifted far from its nominal position, indicated by a tick mark, and has significantly deviated from the defender's centralized belief of the formation's state. As one of the other agents is currently observing it, this discrepancy leads to a large Wasserstein-2 deviation between the updated and predicted belief, triggering a detection.

In comparison, we can look at the behavior of the planned attacker against a planned defense. In Fig. 6.6 below, we see the attacker has remained closer to its nominal position and the centralized belief, however the defender's belief is less accurate for the attacker than the other agents. At this time step, one of its neighbors is observing it, but due to only receiving a relative angle measurement, this will not trigger a detection as the observation the inspecting spacecraft receives is consistent with the defender's belief. However, in the next time step shown in Fig. 6.7, a different neighbor observes the adversary and catches it out of position, raising an alarm.

These qualitative scenarios also demonstrate the need for switching to detect stealth attacks. In each time step shown, the adversary could move arbitrarily far along the line of sight of the agent inspecting it, and no anomaly would be noticed until the formation switches and a different neighbor observed the discrepancy. Similarly, if the adversary had no other spacecraft observing it, it could have moved undetected without restriction, unless the formation reconfigured and observed it.

6.6 Context with Related Work

To address the challenge of stealth attacks, other works propose methods of modifying the system to reveal any currently present stealth attacks [167]–[169], and we have discussed in this chapter different ways of optimizing against several user defined objectives [170]. However, existing approaches are restricted to linear, noise-free systems and specific classes of stealth attacks; namely zero dynamics attacks which leverage output zeros to remain indistinguishable from the unattacked system. Some prior work also considers attacks which provide false information to the centralized monitor, but resolves this through a decentralized approach unaffected by the compromised observations [169].

These approaches to detecting attacks bear resemblance to traditional Fault Detection, Isolation, and Recovery (FDIR) methods [17], [57], including geometric methods which seek to make all possible system faults observable [171]. However, unlike faults, attacks are assumed to be adversarial, seeking to maximize disruption or damage to the system. They are also often restricted to affecting the control inputs, sensing outputs, or communication channels, whereas faults can also result in changes to the system dynamics.

Game-theoretic formulations have previously been proposed to study systems under attack [172]–[174], including methods to defend through design [175] or control of dynamic systems [176]. Here, we extend this approach to switching to detect stealth attacks. Game-theoretic formulation naturally extends to planning both attacks and defenses over a time horizon, as opposed to fixed attacks or single reconfigurations of the sensing topology typically explored in the literature [167]–[170].

6.7 Chapter Summary

We have formalized the problem of constructing and detecting stealth attacks in a multi-spacecraft system which switches observation topologies. By using a metric for the distance between the expected and estimated formation beliefs, we demonstrate the ability to detect anomalous observations which could indicate the presence of false data or hidden attacks in the system. Further, we showed that MCTS based planning methods both enable attackers to maintain stealth with high probability, and defenders to select observation topologies that limit the disturbance an attack can cause while remaining stealthy.

This chapter demonstrates the feasibility of applying game algorithms to consider the complicated problem of stealth attacks. Having done so, there are several opportunities for future work. We assumed an attacker with accurate knowledge of the current defense belief. Work could be done on means to limit the availability of this information to the attacker, or to plan defenses in scenarios where the attacker has even more privileged information, such as knowing what the next defense will be. Similarly, limiting the information available to the defender will make the problem more challenging. In particular, we assumed the defender was aware of which agent would be compromised, if any. One way to adapt our method to the generalized scenario of suspecting all agents would be to perform parallel tree searches for each possibly compromised agent, then select the defense which simultaneously minimizes the expected disruption for all of them. This method proved to be computationally intractable for our currently available resources, but would be enabled by future work to optimize tree searches for partially observable dynamic systems with high dimensional beliefs or hardware advances. Other investigations could apply our method to new defense objectives ranging from maintaining operational performance to avoiding collisions between agents.

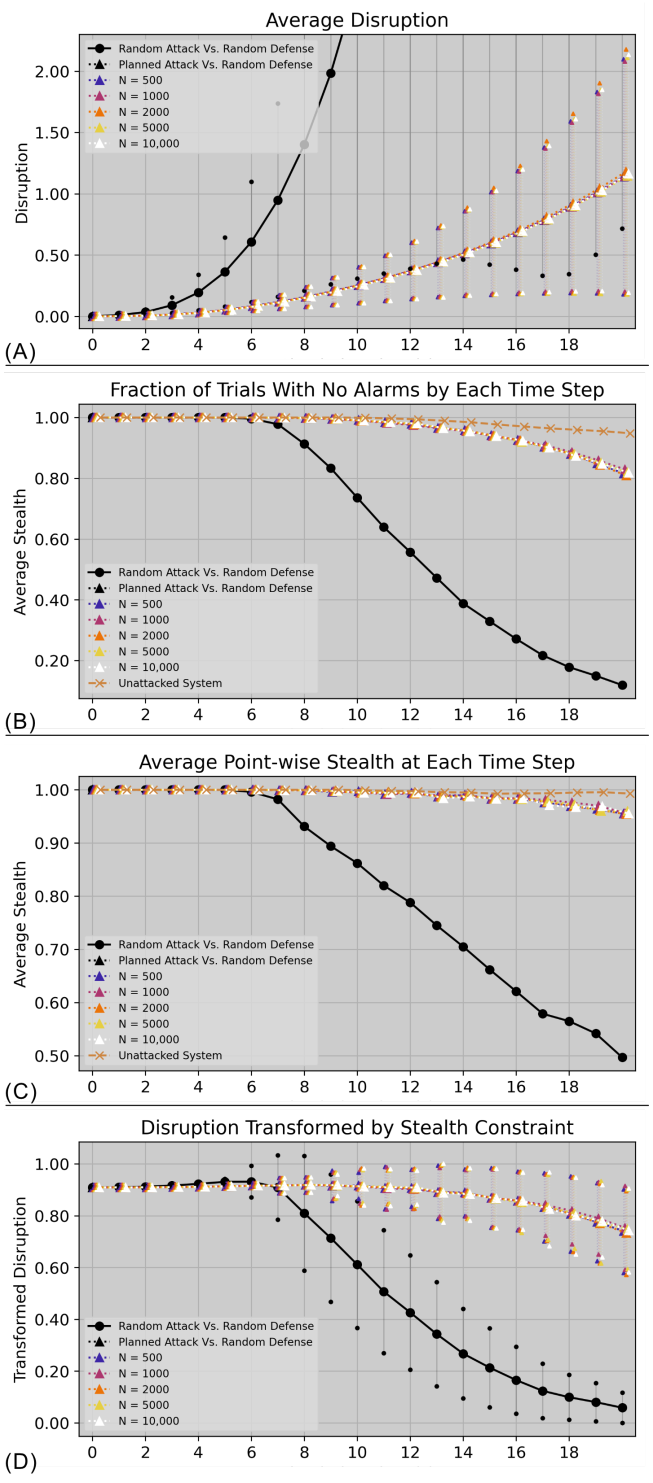


Figure 6.3: **Planned attack vs. random defense.** The disruption, total stealth, point wise stealth and transformed disruption for various levels of attack planning against a random defense. Results are averaged over at least 1000 simulations for each planning level. Note that for readability, the data for each time step is artificially spread out horizontally and $1 - \sigma$ error regions are shown.

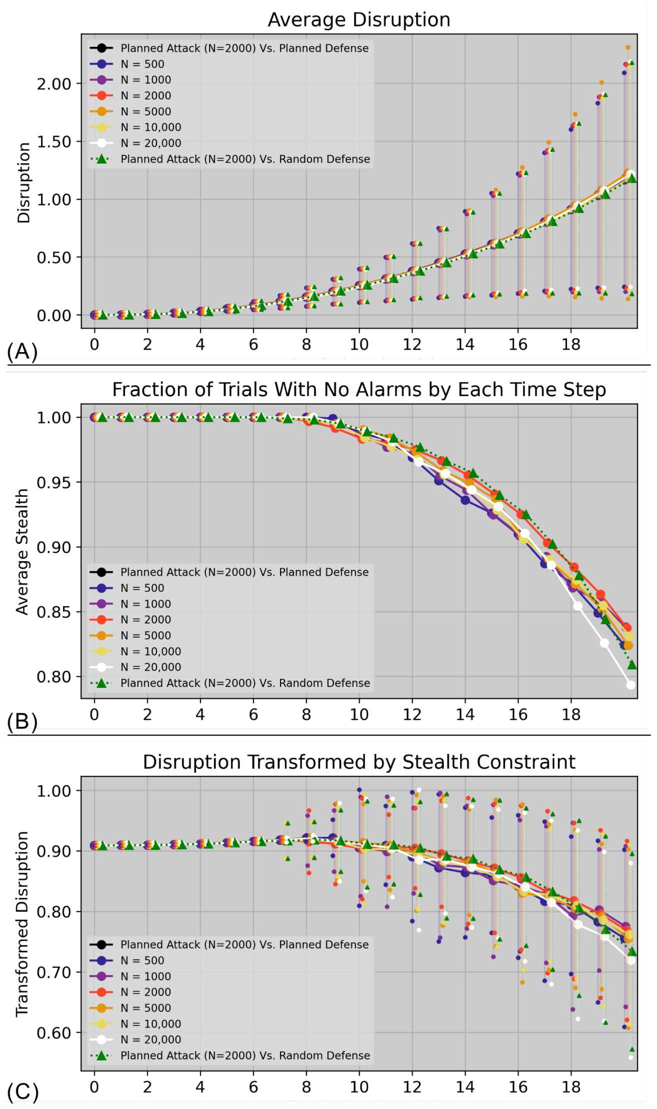


Figure 6.4: **Planned attack vs. planned defense.** The disruption, total stealth, and transformed disruption metric for various levels of defense planning against an attack with constant level of planning ($N = 2000$). Results are averaged over at least 1000 simulations for each planning level. Note that for readability, the data for each time step is artificially spread out horizontally and $1 - \sigma$ error regions are shown.

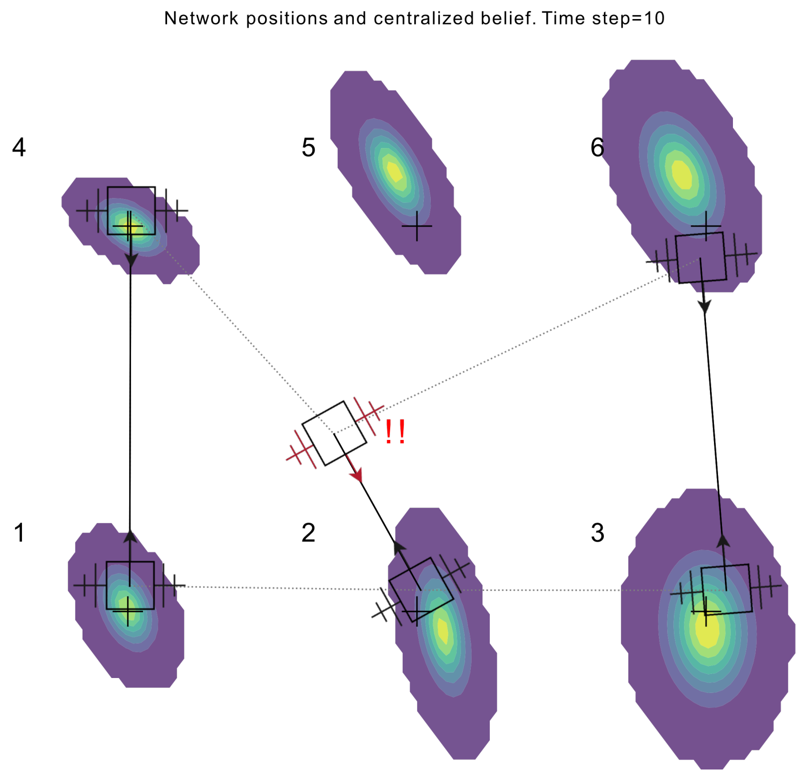


Figure 6.5: **Simulation of a random attacker versus a random defender.** The adversary is the center agent of the top row (agent 5), indicated by red antennas and sensor. The centralized belief of the formation's state is represented as a probability density. Tick marks indicate each agent's nominal position. Spacecraft are capable of looking at their neighbors (indicated by dotted gray connections) and the reference target. Each agent's measurement direction is indicated by an arrow, and a dashed line representing an active sensing link. A solid line indicates the agents are looking at each other. The exclamation points indicate a detection has occurred.

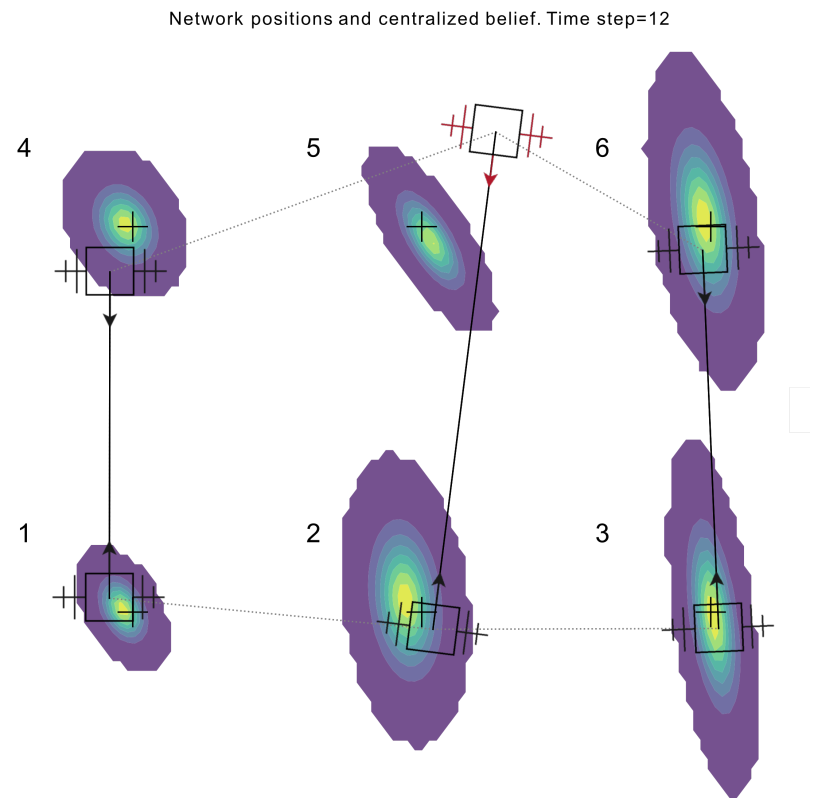


Figure 6.6: **Simulation between a planned** ($N = 2000$) **attacker and a planned** ($N = 20,000$) **defender.** The adversary is the center agent of the top row (agent 5), indicated by red antennas and sensor. The centralized belief of the formation's state is represented as a probability density. Tick marks indicate each agent's nominal position. Spacecraft are capable of looking at their neighbors (indicated by dotted gray connections) and the reference target. Each agent's measurement direction is indicated by an arrow, and a dashed line representing an active sensing link. A solid line indicates the agents are looking at each other.

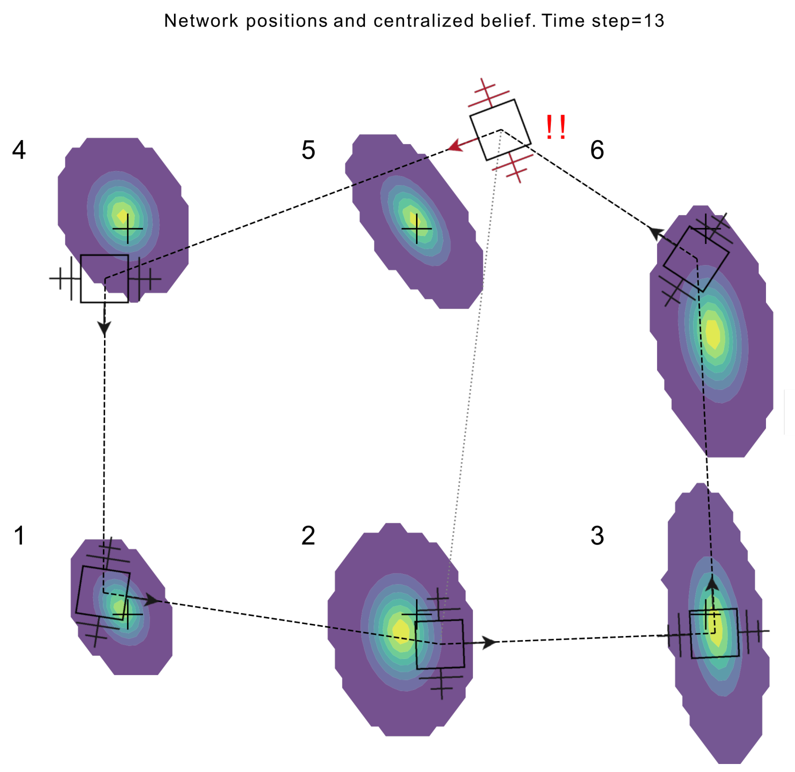


Figure 6.7: **Simulation between a planned** (N = 2000) **attacker and a planned** (N = 20, 000) **defender.** The adversary is the center agent of the top row (agent 5), indicated by red antennas and sensor. The centralized belief of the formation's state is represented as a probability density. Tick marks indicate each agent's nominal position. Spacecraft are capable of looking at their neighbors (indicated by dotted gray connections) and the reference target. Each agent's measurement direction is indicated by an arrow, and a dashed line representing an. A solid line indicates the agents are looking at each other. The exclamation points indicate a detection has occurred.

Chapter 7

EXTENSIONS TO OPTIMAL RELATIVE ORBITS

[1] J. Ragan, R. Ahmed, K. Matsuka, I. Seker, S.-J. Chung, and M. Lavalle, “Optimizing formation flying orbit designs,” *Advances in the Astronautical Sciences AAS/AIAA Spaceflight Mechanics*, vol. 176, 2021. [Online]. Available: `http://www.univelt.com/book=8507`,

[2] J. Lathrop, W. Cook, J. Ragan, and S.-J. Chung, “Applying monte carlo tree search for orbit selection in multi-agent inspection,” *Proceedings of the 2022 AAS/AIAA Astrodynamics Specialist Conference*, 2022. [Online]. Available: `https://www.space-flight.org/docs/2022_summer/ASC22_FullProgram_Compiled.pdf`,

We discuss planning optimal formation flying orbits in this chapter. We consider both online, real-time reconfiguration as well as offline optimization for mission planning. Like the other topics of this thesis, a unifying them is the optimization of novel and non-convex objectives, subject to hard to evaluate constraints.

7.1 Motivation

We have discussed in the previous chapter how formation flying spacecraft have the potential to perform more complex tasks and produce higher quality science while achieving a higher level of redundancy compared to similar monolithic missions in terms of both cost and launch mass, but also introduce new challenges and vulnerabilities. Once such challenge is the increased complexity of orbit design. Each additional spacecraft has its own set of design variables to consider, greatly increasing the size and dimensionality of the parameter space [177].

We study the optimal orbit assignment problem in two cases. First, we consider an Earth observation science mission using Synthetic Aperture Radar (SAR) which was proposed in response to the 2017 Earth Science Decadal Survey [178]. Such missions can provide high resolution 3-D maps of vegetation [179], [180], but present a challenging offline optimization problem. We then consider on-orbit multi-agent inspection, which can enable on-orbit servicing, assembly, and manufacturing [181], external fault detection [182], and interactions with unknown and potentially uncooperative objects [183]. However, the inspection point of interest (POIs) may be unknown in advance or change mid-mission, requiring online orbit assignment.

The remainder of the chapter is organized as follows. In the next section, we review Relative Orbital Mechanics, and TomoSAR and Information Costs. We then consider each of our orbit assignment settings, by formulating the problem, developing our solution, and presenting numerical validations of our method. Finally, we provide a context with other literature and a summary of the chapter.

7.2 Background

This section introduces background material on passive relative orbits, synthetic aperture radar, and information cost metrics. We will use this material to perform our offline optimization for Earth observation science mission and online orbit selection for multi-agent inspections in the next two sections.

Relative Orbital Mechanics

In the design of any formation flying mission, minimizing passive drift is desirable to avoid prohibitive fuel consumption [21]. This has motivated the development of Passive Relative Orbits (PROs), which are formations of spacecraft which remain close under unforced orbital dynamics, eliminating the need for fuel consumption outside of station keeping maneuvers [184]. The simplest and most straight forward method of developing a PRO is to use the linearized mechanics of the Hill-Clohessy-Wiltshire (HCW) equations and eliminate the secular drift terms [185]. However, it neglects both the non-linear dynamics of Keplerian orbital motion and perturbations experienced in low earth orbit (LEO). Assuming each spacecraft have homogeneous drag coefficients, J_2 effects due to Earth's oblatness are the predominant perturbation on relative orbital mechanics. These can be corrected via energy matching conditions, resulting in a nominal drift of as little as 8 millimeters per orbit on kilometer scale formations [186].

To study the evolution of PROs, it is convenient to evaluate the formation in a local reference frame. A common choice is the local vertical, local horizontal (LVLH) frame. This frame is attached to a reference spacecraft designated as the chief, and rotates along the orbit such that the $\hat{x}$ (radial) direction, is radially outward from the chief; the $\hat{z}$ (cross track) direction is along the angular momentum vector of the chief's orbit; and the $\hat{y}$ (along track) direction completes the right-handed coordinate system. In this frame, the remaining spacecraft, known as the deputies, appear to orbit the chief spacecraft with the same period as the formation's orbit around Earth. We will use this frame for the remainder of this chapter. Additional details on computing this frame are provided in Appendix E.

Synthetic Aperture Radar Tomography

Interferometry is a natural science application of formation flying [132], [133]. In this chapter, we focus on SAR applications in particular, in the context of the Distributed Aperture Radar Tomographic Sensors (DARTS) mission concept, part of a NASA Instrument Incubator Program which aims to perform frequent, 3D mapping of Earth's surface topography and vegetation on a near weekly observation cadence [134].

2D imaging radars typically use high bandwidth waveforms and employ the SAR technique of combining multiple radar measurement taken along an orbital trajectory, effectively creating a larger synthetic antenna aperture [187]. These techniques achieve fine range (cross-track) and azimuth (along-track) resolution, but lack depth information such as the vertical structure of vegetation.

Synthetic Aperture Radar tomography (TomoSAR) [179], [188], [189] is a radar imaging technique developed to create three dimensional maps of natural targets. While it is possible to create a radar tomogram by combining 2D images from repeated passes of a single-agent, the DARTS mission concept proposes using multiple formation flying spacecraft to create a synthetic baseline in the cross-track direction instead. This allows for combining numerous 2D slices to create a 3D image in a single orbit pass. The advantages of single-pass TomoSAR imaging are significantly lower temporal decorrelation [190], [191], which can result in poor tomographic accuracy over vegetated areas [192], [193], shorter time to achieve global coverage, and better detection of time varying features. This formation is visualized in Fig. 7.1.

To quantify performance of a TomoSAR system as a function of the spacecraft formation, we consider: (1) the tomographic (vertical) resolution δ_n, and (2) the nearest ambiguity location h_n^a. Tomographic resolution, is inversely proportional to the formation baseline, while the nearest ambiguity location (for uniformly distributed platforms) is inversely proportional to the separation between agents. These quantities are modeled approximately as:

$$\delta_n = \frac{\lambda r_0}{2L_n} \tag{7.1}$$

$$h_n^a = \frac{\lambda r_0}{2\mu_n} \tag{7.2}$$

where L_n is the formation baseline, μ_n is the separation between uniformly distributed platforms, λ is the wavelength of the radar system (assumed to be 24 cm

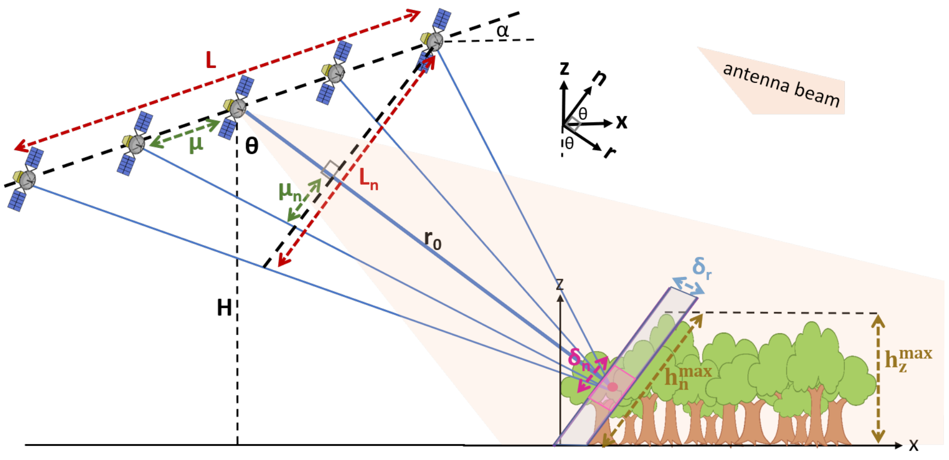


Figure 7.1: **Synthetic Aperture Radar formation diagram.** This formation is flying out of the page, so its velocity vector is omitted. Note that the baseline and and separations between each platform normal to the look direction are the quantities of interest. Here θ denotes the look angle of the formation and defines the radar look axis. α indicates the formation is likely not parallel with the ground. This diagram neglects the curvature of the Earth for simplicity. Reproduced with permission from Seker and Lavalle [194].

(L band) for DARTS), and r_0 is the range to the target. The subscript n indicates projection onto the normal direction perpendicular to both the look and along track directions. Note both these equations assume each platform transmits and receives individually. Other radar modes have a different constant factor. These quantities are visualized in Fig. 7.1. Note that Eq. (7.2) assumes a uniform spacing between the SAR platforms. In our analysis, we use the approximate nearest ambiguity, $\bar{h}_n^a$, computed by replacing μ_n in Eq. (7.2) with the average separation $\bar{\mu}_n$.

To visualize why the nearest ambiguity is an important concern, consider the simulated radar measurements presented in Fig. 7.2. The nearest ambiguity refers to the distance between a main feature of interest, e.g., one of the target points of the simulated array, to the nearest replica of that feature. These replicas are artifacts resulting from the a discrete approximation of the synthetic aperture. Were these signals to overlap, distinguishing them is challenging, if not impossible. This behavior can be seen in the last example of Fig. 7.2, where the platform spacing and formation baseline have both doubled, resulting in overlapping ambiguities. Hence, it is desirable to keep the minimum ambiguity distance greater than the expected target height. For our analysis, a 30 meter constraint was used for good performance on most forests worldwide.

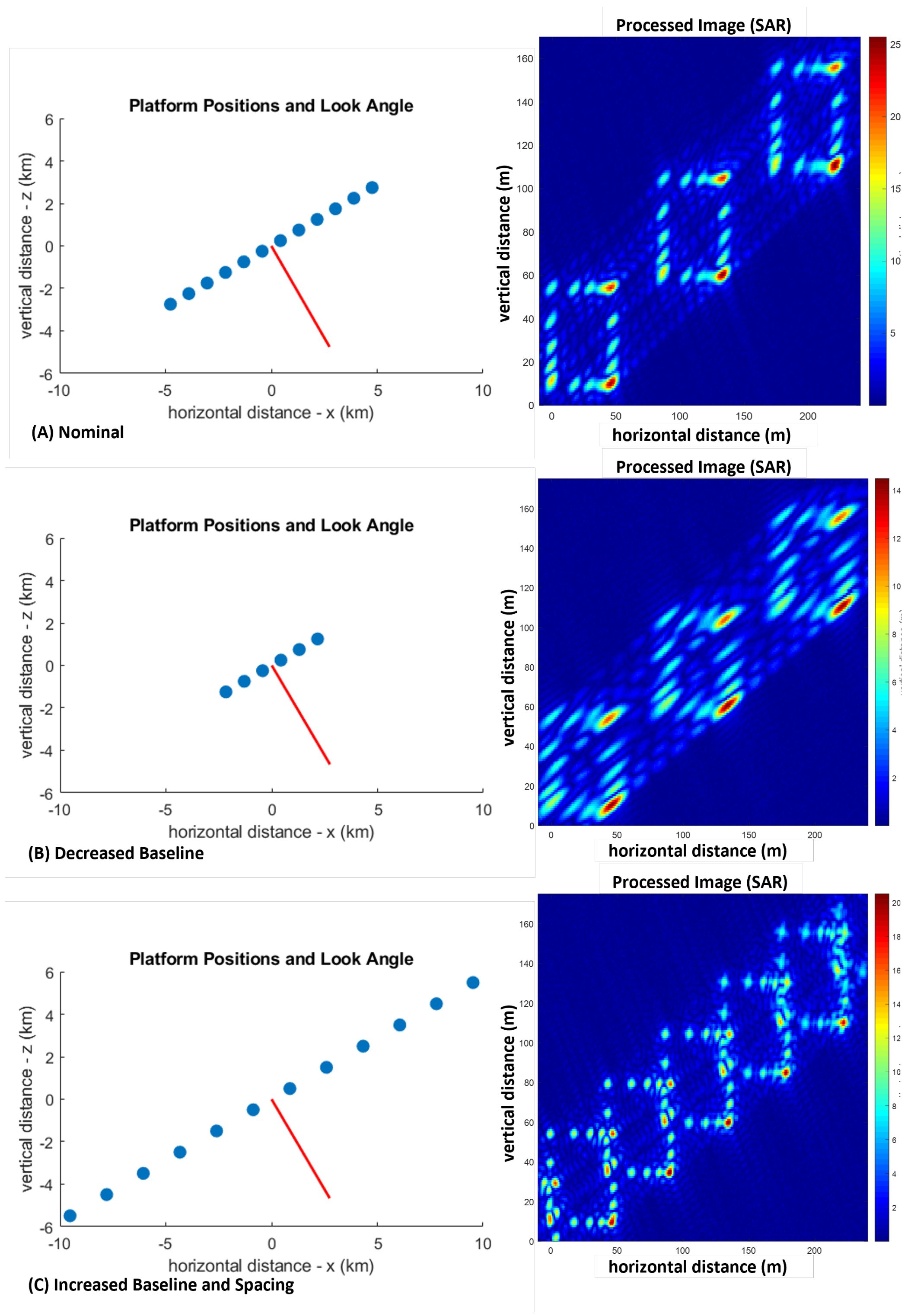


Figure 7.2: **Simulation results adapted from Seker and Lavalle [194].** Depicted are: (**A**) the nominal configuration of the simulation (**B**) decreased formation baseline with unchanged platform spacing and (**C**) doubled baseline and platform separation.

Information Cost for Inspection

To optimize formation flying orbits for inspection tasks, we need a notion of value to quantify how well a formation performs an inspection. This is provided by the information cost H, defined as:

$$H(\mathcal{P}) = \sum_{s \in \text{POIs}} H_{POI}(\mathcal{P}, s) \tag{7.3}$$

$$H_{POI}(\mathcal{P}, s) = \left(w^{-1} + \sum_{p \in \mathcal{P}} f(p, s)^{-1} \right)^{-1} \tag{7.4}$$

$$f(p, s) = \begin{cases} \text{dist}^2(p, s), & \text{if } s \text{ visible from } p \\ \infty, & \text{otherwise} \end{cases} \tag{7.5}$$

Following the notation of Nakka et al. [195], $p \in SE(3)$ is the position and attitude of a sensor on-board a deputy satellite and $s \in \mathbb{R}^3$ is the position of a point of interest onboard the target spacecraft. The function $f(p, s)$ incorporates the observation model between a particular sensor pose p and a point of interest s. In Eq. (7.5), we model the ability of a camera to gather information about a point at a distance as inversely proportional to the squared Euclidean distance.

The information cost H is a function of a set of PRO orbit poses $\mathcal{P}$ and a set of points of interest $s \in$ POIs [195], [196]. It serves as a proxy for the variance of the final estimation error after each observation along the PRO orbits in $\mathcal{P}$. To do so, H sums of the estimation variance of each point of interest s, given an observation contribution $f(p, s)$ and prior variance w.

The information cost of a particular point of interest s is a reciprocal sum of the baseline variance w and the variance of observations made by all deputy satellites throughout their orbit. The set $\mathcal{P}$ in Eq. (7.4) is the set of sensor poses corresponding to every measurement taken from each deputy along their orbits.

The baseline variance w represents the uncertainty in our prior model (if any) of the target. In the limit of w approaching infinity, this means there is zero prior information about the target. In reality, we will always have some slight prior information about the target, and therefore w will be a finite quantity.

7.3 Offline Orbit Design for SAR

We wish to design a formation of spacecraft that provide high resolution TomoSAR images, while maintaining the required ambiguity separation. Like in Chapters 3 and 6, this results in a constrained optimization problem of the form:

$$\begin{aligned} \min_{x_0} \quad & F_f(x_0) + \alpha F_s(x_0) \\ \text{subject to} \quad & \bar{h}_n^a(x_0, t) \geq 30 \forall t \end{aligned} \tag{7.6}$$

where $h_n^a(x_0, t)$ is computed by propagating the relative orbital dynamics, $F_f(x_0)$ is the fuel penalty model and $F_s(x_0)$ is the science objective model given by:

$$F_f(x_0) = \exp\left(\frac{\Delta V_0(x_0)}{\text{ISP}}\right) \tag{7.7}$$

$$F_s(x_0) = -\int_{t_0}^{t_f} \frac{2L_n(x_0, \tau)}{\lambda r_0} \, \mathrm{d}\tau \tag{7.8}$$

where $\Delta V_0(x_0)$ is the largest initial velocity magnitude of the deputy spacecraft required to achieve the J_2 invariant PROs [186], and ISP is the the specific impulse of the agents' propulsion systems. As a result, $F_f(x_0)$ models the fuel mass required by the formation, and F_s models the science value (the reciprocal of the resolution in Eq. (7.1)) integrated along the trajectory from t_0 to t_f. The parameter $\alpha > 0$ is an explicit design parameter that controls the trade off between increased launch mass and increased scientific merit, which is a common trade in space mission designs. If a launch mass budget is specified instead, we could reformulate $F_f(x_0)$ as a constraint in the same fashion as our reformulations in Chapters 3 and 6. Similarly, we can extend this approach to other combinations of science objectives and constraints [134], but will focus on resolution and ambiguity as our examples in this chapter.

Due to both the objective and constraint depending on integrating over the orbital dynamics, the problem defined by Eq. (7.6) is non-convex, making it unsuitable for typical optimization techniques, which require either convexity or the ability to readily convexify the constraints [197], [198]. Instead, like we have done in previous chapters, we will seek numerical methods which provide good approximations of the optimal solution, and can converge to better solutions with increased computational power.

To solve this problem, we employ a genetic algorithm as a heuristic method. Genetic algorithms work by mutating the best currently known solutions to an optimization problem, in our case using Gaussian noise, to seek better related solutions [199]. They are effectively a local minimum search algorithm, which a small chance of mutations escaping to explore other regions of the configuration space. We bias this escape probability in the cross-track direction to seek increases in tomographic baseline, and impose our ambiguity constraint using a barrier function. Our method has the downside of potentially converging to sub-optimal solutions, or requiring many generations to converge, while not providing a closed form solution. However, its advantages are that it imposes no requirements on the objective or constraints and that given a feasible initial configuration it will only improve upon it.

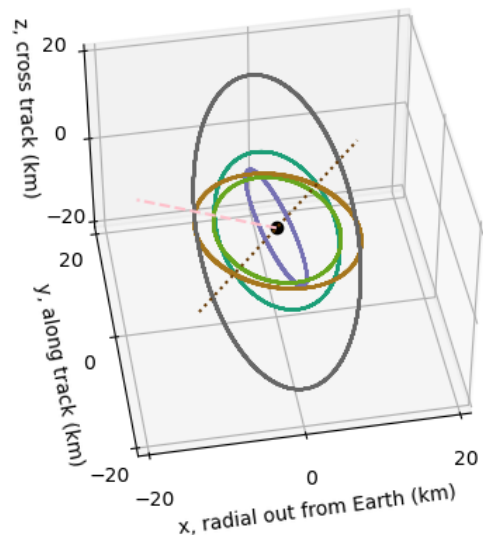
In the next subsection, we present numerical results which validate our algorithm. We note that other design parameters, including the choice of spacecraft, instruments, and the chief orbit (for ground track coverage and frequency of repeated observations) were considered to have been already fixed for the purposes of this analysis.

Simulation Results

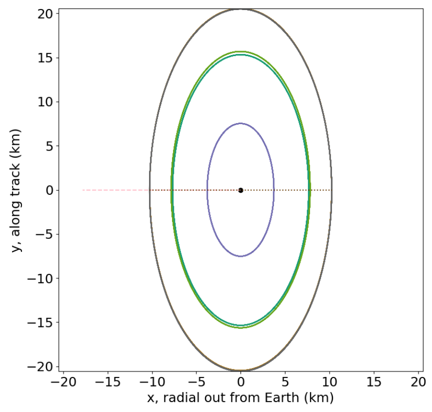
We studied the initial positions converged to after over 100 iterations of our genetic algorithm for formations of six and four spacecraft. The resulting formations are depicted in Figures 7.3 and 7.6, respectively.

Looking at the six spacecraft formation, we note that it achieves a very small drift rate in the along track and radial directions, and stays bounded within the cross track direction. This formation satisfies the ambiguity constraint imposed by the optimization problem, and achieves an average resolution of 7.57 m. The resolution and ambiguity over the first day is shown in Figures 7.4 and 7.5, and is representative of the behavior through the full 12 day period of interest. We note that the 30 m ambiguity constraint is active (there time steps with ambiguity equal to 30 m). This is a well known optimality condition [29], suggesting the genetic algorithm has converged to at least a local minima. The periodic behavior of the resolution and ambiguity match the orbit period, and demonstrate the strong trade off between resolution and ambiguity expected from Eqs. (7.1) and (7.2).

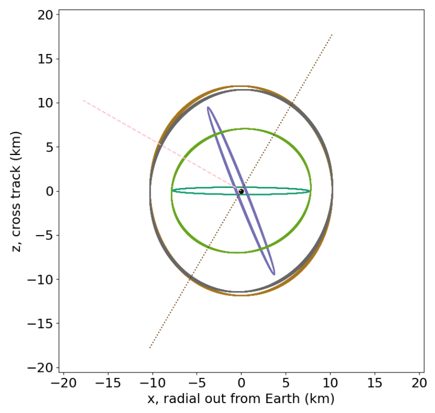
Similar results are produced by the genetic algorithm for 4 spacecraft. The formation is notably more compact to satisfy the ambiguity constraints of the optimization problem. The average resolution of this orbit is 10.82 m, and the average baseline of the formation is 9.72 km. The resulting resolution and ambiguities over the course of the orbit are shown in Figures 7.7 and 7.8. Unsurprisingly, with less agents, a more



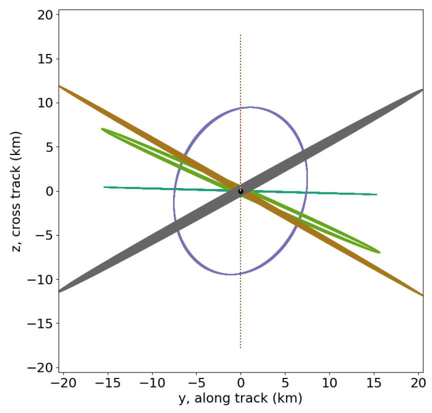
(A) 3D orbit view



(B) Along track vs. radial dynamics



(C) Cross track vs. radial dynamics



(D) Cross track vs. along track dynamics

Figure 7.3: **6 spacecraft (5 deputy) formation.** The formation produced by the genetic algorithm is shown in 3D and 2D views. The look direction of the formation and formation baseline axis are shown in pink (dashed) and brown (dotted), respectively.

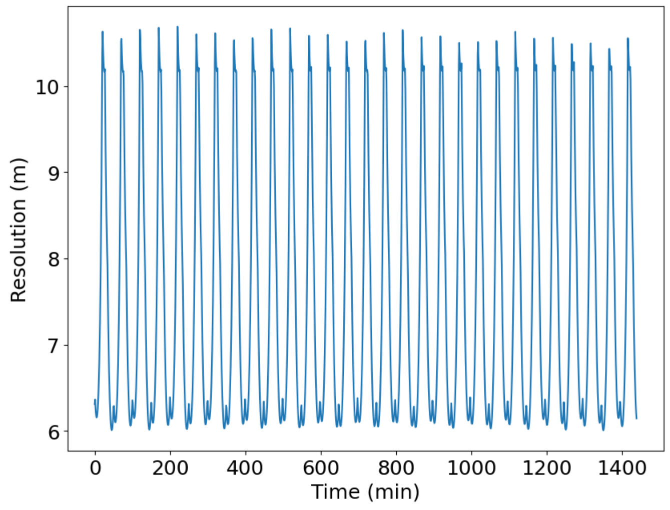


Figure 7.4: **6 spacecraft (5 deputy) formation resolution.** Shown over the period of a day.

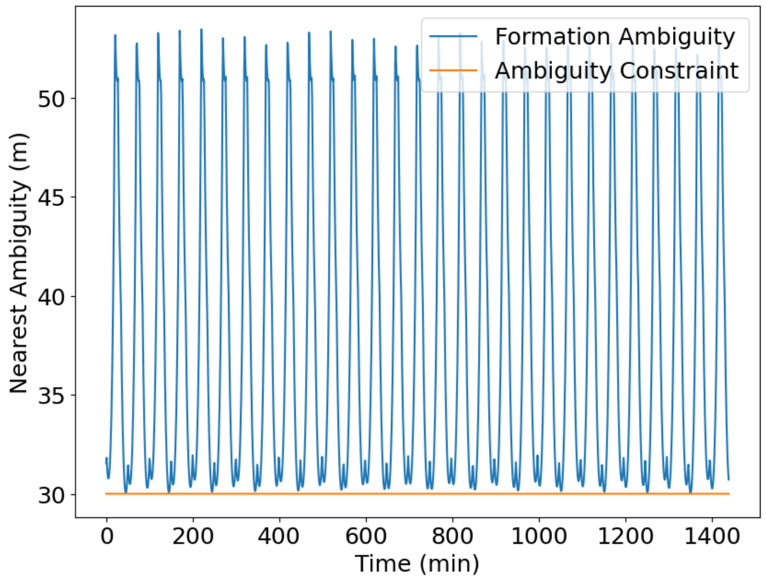
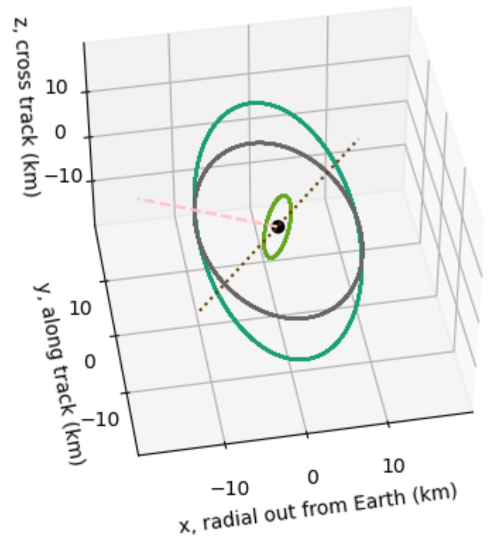
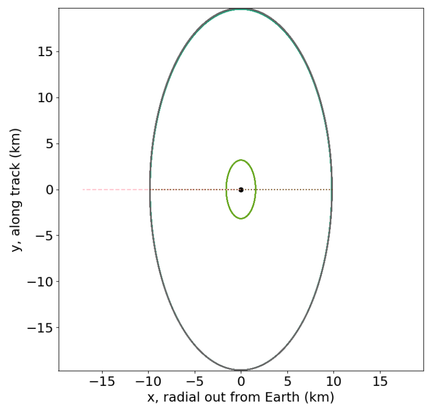


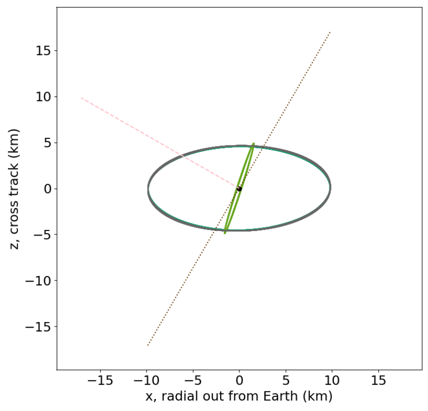
Figure 7.5: **6 spacecraft (5 deputy) formation ambiguity.** Shown over the period of a day. Note the 30 m ambiguity constraint is achieved.



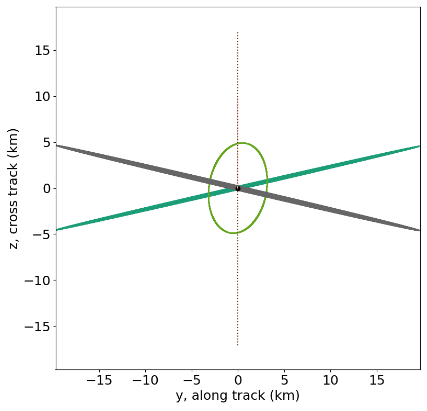
(A) 3D orbit view



(B) Along track vs. radial dynamics



(C) Cross track vs. radial dynamics



(D) Cross track vs. along track dynamics

Figure 7.6: **4 spacecraft (3 deputy) formation.** The formation produced by the genetic algorithm is shown in 3D and 2D views. The look direction of the formation and formation baseline axis are shown in pink (dashed) and brown (dotted), respectively.

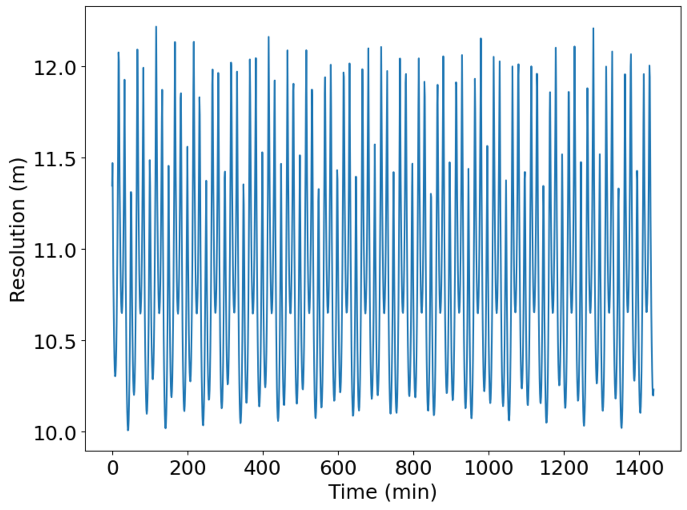


Figure 7.7: **4 spacecraft (3 deputy) formation resolution.** Shown over the period of a day.

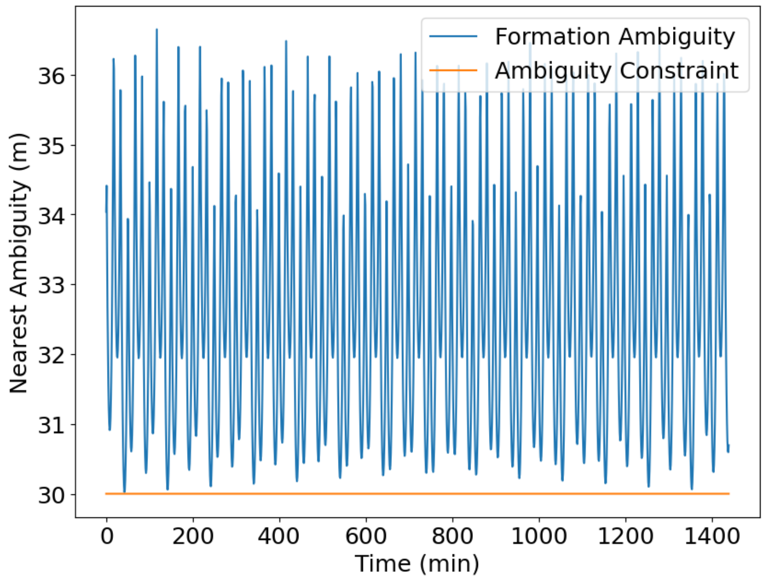


Figure 7.8: **4 spacecraft (3 deputy) formation ambiguity.** Shown over the period of a day. Note the 30 m ambiguity constraint is achieved.

compact formation is needed to maintain a sufficiently high ambiguity separation, and the resolution suffers as a result. We can compare our results with the High Resolution Wide Swath (HRWS) mission concept, which also uses a chief and 3 deputy vehicles to perform SAR observations [200]. Taking a conservative estimate, the HRWS orbit has an average baseline of approximately 800m. Adjusting for differences in frequency and altitude in Eq. (7.1), HRWS achieves the same resolution as our proposed DARTS formation, however does not provide any guarantees of ambiguity constraint satisfaction, validating our algorithm's performance and our additional utility.

7.4 Real-time Orbit Selection for Inspection

Our goal is to find the optimal set of PROs $\mathcal{P}^*$ from a pre-computed library that minimizes the information cost H for a particular target satellite. We denote individual PROs in a library of size n by O_i for $i = 1, ..., n$. We use $\mathcal{P}$ to denote a set of PROs, such as $\mathcal{P} = O_1 \bigcup O_5 \bigcup O_9$ being the collection of PROs 1, 5, and 9 from the library. For a library $C = \{O_1, O_2, ..., O_n\}$ of PROs and a fixed point of interest configuration, the optimal set $\mathcal{P}^*$ is defined as the union of T orbits O_i for T indices $1 \leq i \leq n$ that minimize $H(\mathcal{P})$.

To find the global optimal set of PROs of size T from a library of candidate PROs of size n, we must consider all $\binom{n}{T} = \frac{n!}{T!(n-T)!}$ combinations of PROs. When evaluating a candidate-optimal set of PROs $\mathcal{P}$, the information cost H is found by integrating through the orbits of all T PROs and checking visibility of every point of interest s on the target body every 5 seconds. This evaluation is expensive, and finding the globally optimal solution requires performing it combinatorially many times, making real-time optimization intractable when n is large. Instead, much like we did in Chapters 2, 3, and 6, we will seek out methods that can asymptotically approximate this optimization, and enable any-time solutions.

Asymptotic Approximation as a Sequential Decision-Making Process

Instead of solving for the complete set of PROs at once, which scales combinatorially, we alternatively frame the online assignment of pre-computed PROs to deputy satellites as a sequential decision-making process. This will allow us to employ Monte Carlo Tree Search (MCTS) methods similar to the one used in previous chapters. Now, each decision is the assignment of a new deputy to a pre-computed orbit, with the goal of maximizing the information-gathering ability of the final size T formation. In particular, we will take advantage of MCTS's strength in combinatorial games where the value of early decisions may be unclear until a end state is reached, such as its notable success in Go [125].

Fig. 7.9 visualizes our tree search algorithm. The root node is an empty set, represented by no orbits around the target satellite. The edges below the root node add one PRO, $o_{i,j}$, indicating orbit j is added to a node in layer i. These PROs are represented by the elliptical orbits around the target. Below each child node, the tree continues to extend downward, adding one additional PRO per layer to reach size T.

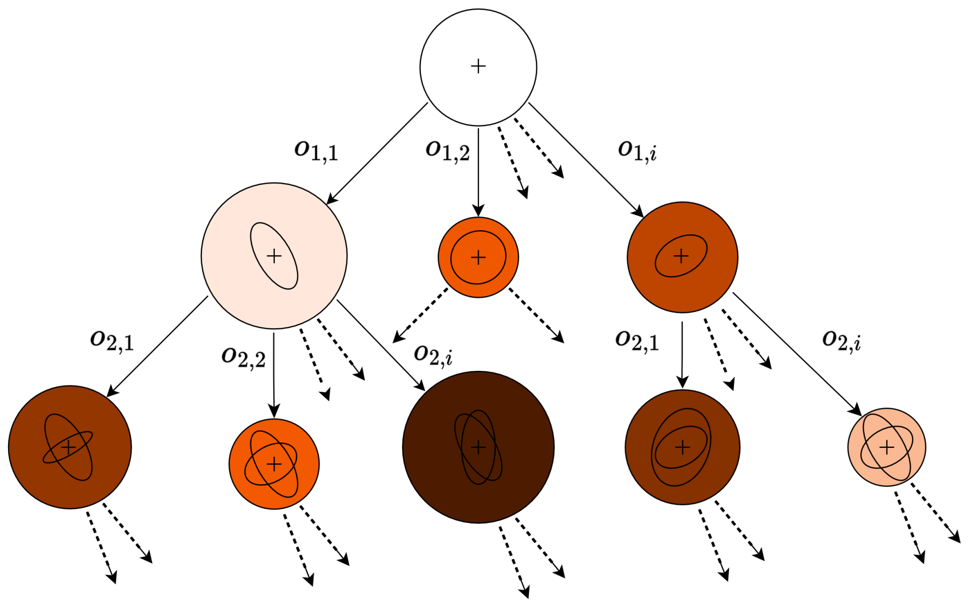


Figure 7.9: **Qualitative visualization of a partially expanded tree search.** Each node of the tree is a set of PROs, represented as orbits around the target. The color of a node indicates the cost (darker is lower) associated with a set of PROs and some points of interest on the target. The size represents the estimated value of each node, based on the average backpropogated reward.

Previous work has explored greedy solutions to this decision making problem [195]. At each step, the next PRO to add to the set is chosen by minimizing the information cost of the added-to set of orbits. This procedure is faster than a global optimum search, as T decisions are to be made, and each requires n evaluations or fewer, yielding $O(nT)$ evaluations total.

This approach, while fast, fails to plan long-term, and we will show this can lead to sub-optimal orbit designs. One reason for this is the inability to wait until all orbits are chosen to evaluate the formation cost. Instead, the greedy algorithm requires cost evaluations with incomplete sets of PROs containing fewer than T orbits. In situations with little prior information, this can yield extremely large information costs (H). If a single orbit cannot see every point of interest on the target, the full contribution w from each unseen point is added to the information cost, representing the high uncertainty remaining. In the limit $w \to \infty$, the greedy approach will fail, as the cost of every first choice of PRO will be ∞. In the example illustrated in Fig. 7.9, the optimal solution for a two orbit formation is the darkest node at the bottom of the dree. However, a greedy policy would select the right branch of the tree at its first step, missing the most-optimal solution.

Simulation Overview

We provide an overview of the simulations used to validate our algorithm in this section. Further details are provided in Appendix E. For each of our experiments, we consider one of the three points of interest configurations show in Fig. 7.10. These consist of: six points uniformly distributed on a sphere, an asymmetric target with a cluster of points on one side, and fifty points evenly distributed on the target satellite.

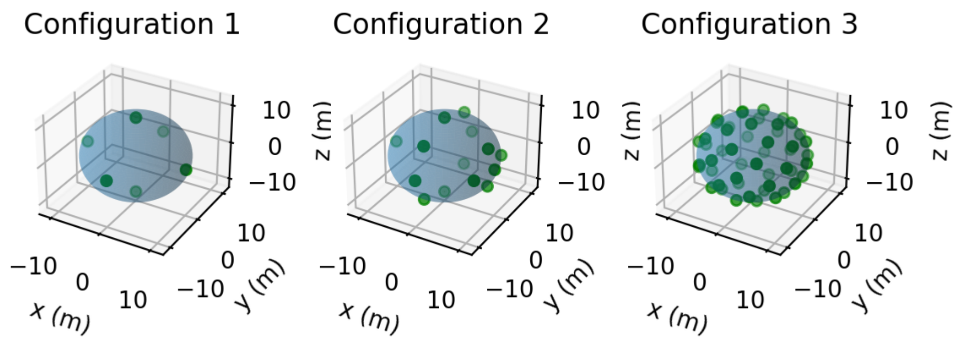


Figure 7.10: **The three Point of Interest configurations studied (LVLH coordinates).** In the first configuration, (left) six points of interest are placed at the positive and negative extremes of each axis. In the second configuration (center), more points of interest are placed on the positive-x hemisphere and meridian separating the positive- and negative-x hemispheres. This configuration represents a target satellite with more components of interest on one side. In the third configuration (right), 50 points of interest are evenly distributed across the target. This configuration represents a target satellite over which complete coverage is desired.

We first search over small libraries where the optimal formation can be tractably computed through exhaustive search ($n = 8, ..., 20$, $T = 4$) to show our algorithm converges towards the optimal formation while the greedy algorithm quickly becomes sub optimal. We then consider large libraries and formations ($n = 80, 100, 120, 140, 160,\ T = 10$) where the optimal formation cannot be readily found, to show we maintain a performance gap as the problem scales. In all formations, we construct the PROs by selecting initial positions uniformly distributed on a sphere centered on the inspection target, at radii of approximately 1 km.

Unlike the ray-casting based visibility check for the evaluation of $f(p, s)$ used in pervious work [195], we consider a visibility cone method to evaluate $f(p, s)$ along each PRO and assume the sensor is always pointed at the target. We model the prior information with a baseline variance of $w = 100$ for all points.

Our MCTS algorithm is similar to the versions presented in Chapters 2 and 3, with a slight modification to prevent continued selection of fully expanded branches in small libraries and formations. Pseudocode and details of our algorithm are also included in Appendix E.

Simulation Results

In every scenario, the MCTS algorithm outperformed the greedy algorithm in information cost. The cost for the three point of interest configurations are shown in Fig. 7.11. This figure shows the cost evaluated for PRO libraries of size varying from $n = 8$ to $n = 20$. Each point of interest can contribute up to $w = 100$ to the total information cost, which is then normalized by the number of points of interest in a configuration. The value in Fig. 7.11 represents un-gathered information.

In Configuration 1, assigning four orbits to observe six points of interest allows very good coverage of the points, with 65% of the information about each point being gathered on average by our method and the optimal solution. The clustering of points in Configuration 2 allows even better coverage as, near-pass observations of multiple points can be made with a single orbit, whereas the 50 uniformly-spread points in Configuration 3 cannot be efficiently inspected by 4 orbits. Notably, even in the third configuration, the MCTS solution typically matches the optimal solution exactly, and always outperforms the greedy policy.

MCTS especially outperforms the greedy assignment policy when the points of interest are not uniformly distributed on the target, as in POI configuration 2. Here, MCTS always finds a set of orbits with a cost within 20% of the optimal cost while the greedy policy finds solutions 30% to 270% worse than the optimal. The resulting formations for the greedy, MCTS, and optimal policies are shown in Fig. 7.12 for a PRO library of size $n = 12$ and the second POI configuration.

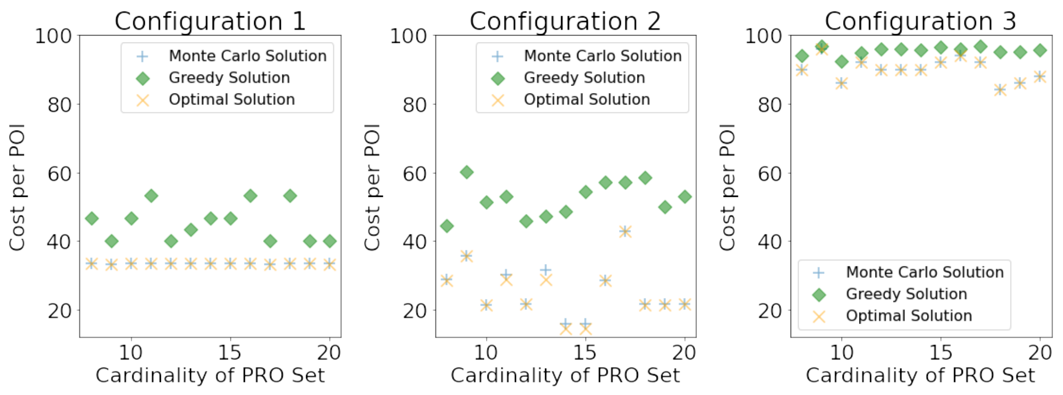


Figure 7.11: **The information cost of greedy and MCTS orbit assignment policies.** In each graph, the size of the PRO library used in the simulation is the x-axis, and the normalized information cost is shown on the y-axis. The point of interest configurations are six uniform points, one hemisphere containing more points, and fifty uniform points.

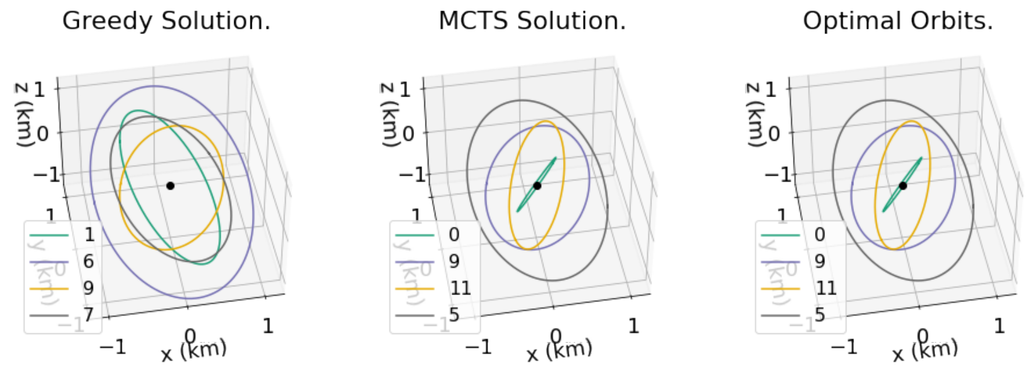


Figure 7.12: **Resulting PRO formations.** Shown are all three studied orbit assignment policies for the second POI configuration and library of size $n = 12$. The MCTS policy agrees with the optimal solution of PROs with indices $\{0, 9, 11, 5\}$. The greedy solution found a sub-optimal arrangement of PROs, $\{1, 6, 9, 7\}$. For this simulation, the information cost of the optimal and MCTS sets is $H = 303$ and the information cost of the greedy set is $H = 642$ (a factor of 2.12 worse). The target is indicated by the black dot in the center of each figure.

We also show orbit solutions found for the larger library sizes. The left plot of Fig. 7.13 shows the information cost per POI for the large library sizes applied to Configuration 3, where the small libraries performed the worst. The global optimum was not solved for and is not shown.

The orbit solutions shown in center and right plots of Fig. 7.13 showcase the weakness of the greedy approach. The greedy policy selects the smallest central

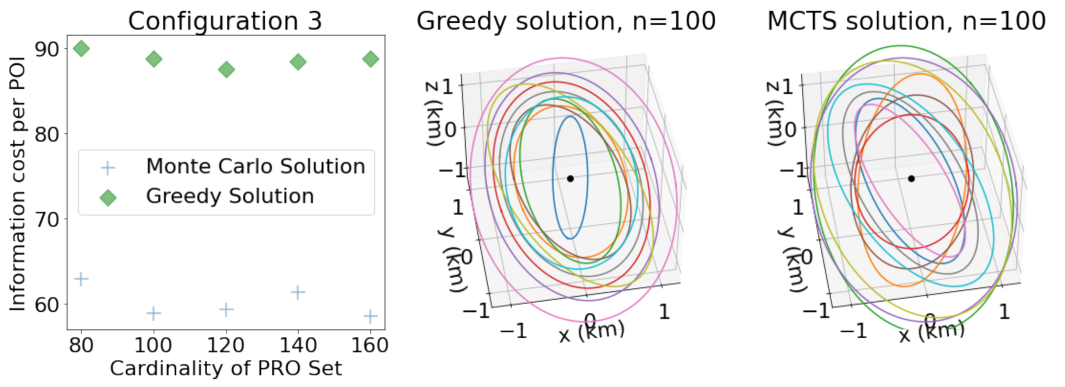


Figure 7.13: **Resulting Information cost and representative orbit solutions for $n = 80, 100, 120, 140, 160$.** On the left, the information cost of the greedy and MCTS policies are compared. The optimal solution was not found for these library sizes. The center and right images are the orbit solutions found for $n = 100, T = 10$ by the greedy and MCTS policy, respectively. For this simulation, the information cost of the MCTS set is $H = 2944$, 58.8 per POI, and the information cost of the greedy set is $H = 4441$, 88.82 per POI (a factor of 1.51 worse). The target is indicated by the black dot in the center of each figure.

orbit (in blue) as the first choice of orbit, as it performs a close pass to many points. This choice results in the final set being sub-optimal, as the points of interest can be more optimally observed through a set of PROs further away from the target. The MCTS policy finds this set whose information cost is a factor of 1.51 better.

On even the largest ($n = 160$,$T = 10$) libraries and formations, our MCTS algorithm balances well the trade off between rapid execution and performance. In this setting, the MCTS algorithm takes 148 s to complete, compared to 89 s for the greedy algorithm and an estimated 4.14 million years for a naive brute force evaluation. Even in the smaller formations, it took up to 278 s for a brute force evaluation of the $n = 20$ library, where as MCTS ran in 5.74 s and recovered the optimal solution.

Finally in Fig. 7.14, we show the information cost averaged over five random seeds of the as-yet best set of PROs found versus iteration of the Monte-Carlo tree search policy. The orbit assignment tasks shown are the same as shown in Fig. 7.12 ($n = 16$, POI Configuration 2) and Fig. 7.13 ($n = 100$, POI Configuration 3). In the case of a small library size (left plot), every instance of the MCTS algorithm arrives at the optimal set of PROs by iteration 45. In the right plot, the MCTS cost continues to decrease until the search is terminated at iteration $M = 1000$.

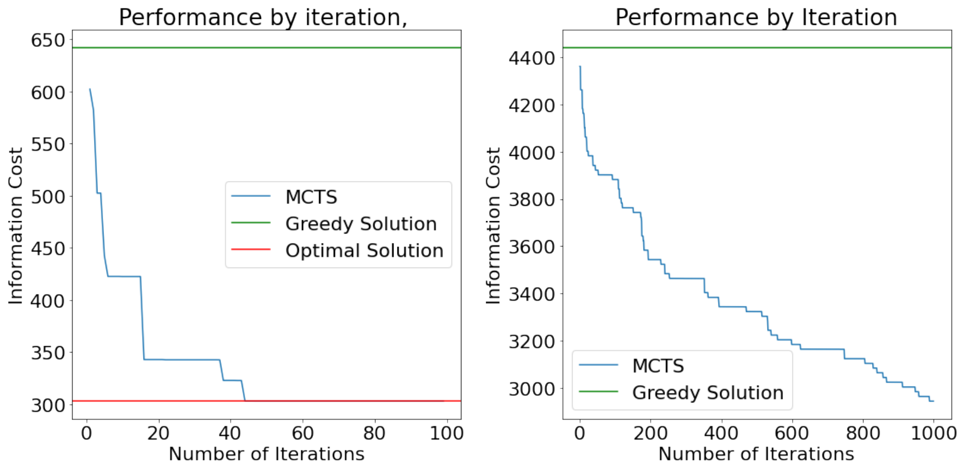


Figure 7.14: **The current-best set performance over MCTS iteration.** The left plot is for POI Configuration 2, library $n = 12$, the same simulation as in Fig. 7.12. The performance of the optimal policy and the greedy policy are shown as red and green lines, respectively. By iteration 45, the tree search has arrived at the optimal solution. The right plot corresponds to POI Configuration 3, library $n = 100$, the same simulation as in Fig. 7.13. Only the MCTS and greedy costs are shown, as the optimum was not computed for this library. Both plots are averaged across five random seeds.

7.5 Context with Related Work

Offline Orbit Design for SAR

With the success of the TanDEM-X mission in 2010, [201] several follow up formation flying synthetic aperture radar (SAR) missions have been proposed in previous work, including its candidate successor Tandem-L [202]. Proposed advances to existing formation flying SAR missions include adapting the Sentinel-1 mission in similar manner as TadDEM-X by adding two receive-only companions [203], [204], or modifying the TandDEM-X mission design to include up to four companion satellites [200], [205]. However, no formation of 6 or more spacecraft has been proposed.

Optimizing formation flying trajectories is also a widely studied research topic [206]–[210], including research into optimal formation reconfigurations via analytic methods [210] as well as genetic algorithms [206]. Previous formation flying optimization work have optimized the formation stability [211] or fuel costs due to ΔV (velocity change) [212], but have not considered a science objective function or applied optimization techniques to an orbital TomoSAR formation.

Real-time Orbit Selection for Inspection

Previous work have included the design of estimators for multi-agent inspection [135], offline optimization of orbits for visual inspection [136], greedy assignment using orbits designed offline [213], and deep-learning for online orbit reconfiguration [214]. Our method extends this work by providing a new search based methodology, extending the strengths of MCTS algorithms demonstrated in pervious chapters to the real-time orbit assignment problem.

7.6 Chapter Summary

In this chapter, we have extended our work on optimal planning to the domain of formation flying orbit design through consideration of two problems. In the first, we considered offline optimization of a formation of TomoSAR spacecraft to maximize a science output matrix while satisfying constraints to ensure usable measurements. Using a genetic algorithm, we developed formations for four and six spacecraft, demonstrating the utility and feasibility of our method. This approach could be extended to other science driven metrics or constraints, or other missions where performance is a direct function of relative orbital dynamics.

The second problem we considered demonstrated how our real-time search algorithms could be deployed to reconfigure an existing swarm of spacecraft to achieve inspection tasks. By using MCTS to efficiently search the design space, we were able to reliably outperform a greedy method, and do so in a fraction of the time needed for exhaustive brute force searches. Combined with the extensions to information gathering problems and probabilistic constraints explored in previous chapters, this work demonstrates the broad applicability of MCTS planning methods in domains where a trade off between optimality and real-time performance is desirable.

Chapter 8

CONCLUSION

Over the course of this thesis, we have employed MCTS and other planning algorithms to solve a range of emerging challenges for space missions. We have shown that careful formalizing of the problems of interest can result in forms that our methods can efficiently search over while provably converging to optimal solutions. In Chapter 2, this approach enabled us to perform and scale better in information gathering problems than existing partially observable methods. In Chapter 3, it enabled us to further extend our method to include chance constraints which can provide guarantees of safety under mild conservative approximations. In Chapter 4, we discussed why our methodology was needed for information gathering problems, and discussed our applicability to other systems and real-time scenarios.

Our work in active fault estimation then motivated us to consider new methods of MCTS planning, creating in Chapter 5 an array-based alternative to the conventional implementations which feature predictable branching. We also considered extensions to multi-agent systems, both to defend against adversarial stealth attacks in Chapter 6, and to plan optimal orbital configurations in Chapter 7.

A unifying theme of this thesis has been using planning to act optimally in the face of uncertain future information. This uncertainty has come from faults, external actors, and complexity. Often, the number of possible futures has been far too large to exhaustively search over. However, by efficiently balancing exploring new possibilities with focusing our attention on areas with high probabilities of success and reward, we have shown that autonomous agents can act intelligently in ways that can be readily interpreted by human operators. In doing so, our research has also presented several promising opportunities for future work.

8.1 Future Work

Continuous Fault Models

In our development of FEAST and s-FEAST in Chapters 2 and 3, we considered enumerated faults selected by an operator based on faults seen in prior events, modeling of likely fault scenarios, or chosen to monitor particularly dangerous failures. This allowed us to condition our marginalized filter on each fault scenario, and efficiently decompose the belief over state and fault space. By assuming a structure of the faults, we were also able to successfully avoid observability concerns which can arise when there is no direct sensing of the faults or too general of an input-output model. However, the trade off is that a finite, enumerated set of faults may not be suitable to scenarios where unknown or unmodeled faults dominate, or where slight differences between the modeled and actual faults lead to large differences in observations. Approaches we considered to address this gap include modeling the enumerated faults as particles and leveraging techniques from particle filtering such as resampling or diffusion to explore the fault space. Alternatively, the enumerated faults could be instead formulated as a continuous region of similar faulty behavior instead of a single point in fault space, and observations consistent with any fault in these regions could be equally weighted.

Improved MCTS Performance in Dynamical Systems

Our work to create an implementation of Monte Carlo Tree Search using arrays instead of tree objects to store the search data has presented promising preliminary results. However, there remains plenty to be done to further optimize this method. One particular area of interest is taking advantage of the algorithm's avoidance of branch prediction to deploy our method on hardware accelerators which have limited branching capabilities, such as GPUs. Thorough profiling of our algorithm could also be performed, to identify areas where more optimization is possible, and to validate our intuition about the importance of the processor cache to our method's performance. Additional algorithmic optimizations can be performed, especially by taking advantage of domain or problem specific knowledge to identify fully expanded layers of the tree and tighter bounds on state branching. Finally, we predict that this approach would pair well with ongoing efforts to parallelize MCTS, and may open new avenues to do so.

BIBLIOGRAPHY

[1] Y. Gao and S. Chien, "Review on space robotics: Toward top-level science through space exploration," *Science Robotics*, vol. 2, no. 7, eaan5074, 2017. DOI: `10.1126/scirobotics.aan5074`. eprint: `https://www.science.org/doi/pdf/10.1126/scirobotics.aan5074`. [Online]. Available: `https://www.science.org/doi/abs/10.1126/scirobotics.aan5074`.

[2] J. Balaram, M. Aung, and M. P. Golombek, "The ingenuity helicopter on the perseverance rover," *Space Science Reviews*, vol. 217, no. 4, p. 56, 2021.

[3] Y. Li, H. Li, W. Liu, *et al.*, "A networking perspective on starlink's self-driving leo mega-constellation," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, ser. ACM MobiCom '23, Madrid, Spain: Association for Computing Machinery, 2023, ISBN: 9781450399906. DOI: `10.1145/3570361.3592519`. [Online]. Available: `https://doi.org/10.1145/3570361.3592519`.

[4] S. Nag, D. D. Murakami, N. A. Marker, M. T. Lifson, and P. H. Kopardekar, "Prototyping operational autonomy for space traffic management," *Acta Astronautica*, vol. 180, pp. 489–506, 2021.

[5] F. Rossi, D. A. Allard, R. Amini, *et al.*, "Workflows. user interfaces, and algorithms for operations of autonomous spacecraft," in *2023 IEEE Aerospace Conference*, IEEE, 2023, pp. 1–17.

[6] T. Uhlig, F. Sellmaier, and M. Schmidhuber, *Spacecraft operations*. Springer, 2015.

[7] I. A. Nesnas, L. M. Fesq, and R. A. Volpe, "Autonomy for space robots: Past, present, and future," *Current Robotics Reports*, vol. 2, no. 3, pp. 251–263, 2021.

[8] P. S. Morgan, "Fault protection techniques in JPL spacecraft," version V2, 2005. DOI: `2014/39531`. [Online]. Available: `https://hdl.handle.net/2014/39531`.

[9] A. Nelessen, C. Sackier, I. Clark, *et al.*, "Mars 2020 entry, descent, and landing system overview," in *2019 IEEE Aerospace Conference*, 2019, pp. 1–20. DOI: `10.1109/AERO.2019.8742167`.

[10] A. Johnson, S. Aaron, J. Chang, *et al.*, "The lander vision system for Mars 2020 entry descent and landing," *Guidance, Navigation, and Control 2017*, vol. 159, no. JPL-CL-CL# 17-0445, 2017.

[11] C. Pardini and L. Anselmo, "Evaluating the impact of space activities in low earth orbit," *Acta Astronautica*, vol. 184, pp. 11–22, 2021, ISSN: 0094-5765. DOI: `https://doi.org/10.1016/j.actaastro.2021.03.030`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0094576521001430`.

[12] K. Howard and A. Von Ah, "Large constellations of satellites: Mitigating environmental and other effects," *Government Accountability Office, Washington, DC, USA, GAO Report No. GAO-22-105166*, 2022.

[13] B. Balaram, T. Canham, C. Duncan, *et al.*, "Mars helicopter technology demonstrator," in *2018 AIAA Atmospheric Flight Mechanics Conference*, 2018, p. 0023.

[14] W. S. Slater, N. P. Tiwari, T. M. Lovelly, and J. K. Mee, "Total ionizing dose radiation testing of NVIDIA Jetson Nano GPUs," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 2020, pp. 1–3.

[15] P. R. Turner, "Command and telemetry in autonomous spacecraft design," in *International Telemetering Conference Proceedings*, International Foundation for Telemetering, vol. 20, 1984.

[16] R. Rudd, J. Hall, and G. Spradlin, "The voyager interstellar mission," *Acta Astronautica*, vol. 40, no. 2-8, pp. 383–396, 1997.

[17] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 636–653, 2009.

[18] A. Wander and R. Förstner, *Innovative fault detection, isolation and recovery strategies on-board spacecraft: state of the art and research challenges*. Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV Bonn, Germany, 2013.

[19] M. McIntyre, W. Dixon, D. Dawson, and I. Walker, "Fault detection and identification for robot manipulators," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 5, 2004, 4981–4986 Vol.5. DOI: `10.1109/ROBOT.2004.1302507`.

[20] S. Bandyopadhyay, R. Foust, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Review of formation flying and constellation missions using nanosatellites," *Journal of Spacecraft and Rockets*, vol. 53, no. 3, pp. 567–578, 2016. DOI: `10.2514/1.A33291`. eprint: `https://doi.org/10.2514/1.A33291`. [Online]. Available: `https://doi.org/10.2514/1.A33291`.

[21] D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen, "A survey of spacecraft formation flying guidance and control. part II: Control," in *Proceedings of the 2004 American Control Conference*, IEEE, vol. 4, 2004, pp. 2976–2985.

[22] D. C. Schedl, I. Kurmi, and O. Bimber, "An autonomous drone for search and rescue in forests using airborne optical sectioning," *Science Robotics*, vol. 6, no. 55, eabg1188, 2021.

[23] V. Verma, M. W. Maimone, D. M. Gaines, *et al.*, "Autonomous robotics is driving Perseverance rover's progress on Mars," *Science Robotics*, vol. 8, no. 80, eadi3099, 2023.

[24] T. Ishigooka, S. Honda, and H. Takada, "Cost-effective redundancy approach for fail-operational autonomous driving system," in *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*, 2018, pp. 107–115. DOI: 10.1109/ISORC.2018.00023.

[25] A. Mantooth, C.-M. Zetterling, and A. Rusu, "Venus calling silicon carbide radio circuits can take the heat needed to phone home from our hellish sister planet," *IEEE Spectrum*, vol. 58, no. 5, pp. 24–30, 2021.

[26] S. A. Jacklin, "Small-satellite mission failure rates," NASA Ames Research Center, Tech. Rep., 2019.

[27] F. A. Authority, *FAA aerospace forecast: Fiscal years 2019-2039*, 2019.

[28] M. Osborne, J. Lantair, Z. Shafiq, *et al.*, "UAS operators safety and reliability survey: Emerging technologies towards the certification of autonomous UAS," in *2019 4th International Conference on System Reliability and Safety (ICSRS)*, 2019, pp. 203–212. DOI: 10.1109/ICSRS48664.2019.8987692.

[29] D. E. Kirk, *Optimal control theory : an introduction*. Courier Corporation, 2004.

[30] M. J. Kochenderfer, *Decision Making Under Uncertainty: Theory and Application*. The MIT Press, Jul. 2015.

[31] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics* (Intelligent robotics and autonomous agents). MIT Press, 2005.

[32] J. Speyer, J. Deyst, and D. Jacobson, "Optimization of stochastic linear systems with additive measurement and process noise using exponential performance criteria," *IEEE Transactions on Automatic Control*, vol. 19, no. 4, pp. 358–366, 1974. DOI: 10.1109/TAC.1974.1100606.

[33] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[34] C. Browne, E. J. Powley, D. Whitehouse, *et al.*, "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012. DOI: 10.1109/TCIAIG.2012.2186810.

[35] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, Springer, 2006, pp. 282–293.

[36] D. Shah, Q. Xie, and Z. Xu, "Non-asymptotic analysis of Monte Carlo tree search," in *SIGMETRICS (Abstracts)*, ACM, 2020, pp. 31–32.

[37] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," *Advances in neural information processing systems*, vol. 23, pp. 2164–2172, 2010.

[38] G. Süssmann, "Uncertainty relation: From inequality to equality," *Zeitschrift für Naturforschung A*, vol. 52, no. 1-2, pp. 49–52, 1997.

[39] R. G. Brown and P. Y. Hwang, "Introduction to random signals and applied kalman filtering: With matlab exercises and solutions," *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions*, 1997.

[40] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "FastSLAM: a factored solution to the simultaneous localization and mapping problem," *Eighteenth National Conference on Artificial Intelligence*, vol. 593598, 2002.

[41] M. Kontitsis, E. A. Theodorou, and E. Todorov, "Multi-robot active SLAM with relative entropy optimization," in *2013 American Control Conference*, 2013, pp. 2757–2764. DOI: 10.1109/ACC.2013.6580252.

[42] J. Marino, M. Cvitkovic, and Y. Yue, "A general method for amortizing variational filtering," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018.

[43] Y. K. Nakka, R. C. Foust, E. S. Lupu, *et al.*, "A six degree-of-freedom spacecraft dynamics simulator for formation control research," in *AAS/AIAA Astrodynamics Specialist Conference*, AIAA, 2018.

[44] R. Foust, E. Lupu, Y. Nakka, S.-J. Chung, and F. Hadaegh, "Autonomous in-orbit satellite assembly from a modular heterogeneous swarm," *Acta Astronautica*, vol. 169, pp. 191–205, Jan. 2020, ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2020.01.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0094576520300060.

[45] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[46] J. G. Saw, M. C. Yang, and T. C. Mo, "Chebyshev inequality with estimated mean and variance," *The American Statistician*, vol. 38, no. 2, pp. 130–132, 1984.

[47] A. Kabán, "Non-parametric detection of meaningless distances in high dimensional data," *Statistics and Computing*, vol. 22, pp. 375–385, 2012.

[48] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation.," in *Robotics: Science and Systems*, Cambridge, MA, USA, vol. 13, 2017, pp. 1–10.

[49] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, “Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming,” *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[50] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, “Model predictive control of swarms of spacecraft using sequential convex programming,” *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 6, pp. 1725–1740, 2014.

[51] Y. K. Nakka and S.-J. Chung, “Trajectory optimization of chance-constrained nonlinear stochastic systems for motion planning under uncertainty,” *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 203–222, 2022.

[52] R. Cosner, P. Culbertson, A. Taylor, and A. Ames, “Robust Safety under Stochastic Uncertainty with Discrete-Time Control Barrier Functions,” in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, Jul. 2023. DOI: 10.15607/RSS.2023.XIX.084.

[53] T. A. N. Heirung and A. Mesbah, “Input design for active fault diagnosis,” *Annual Reviews in Control*, vol. 47, pp. 35–50, 2019, ISSN: 1367-5788.

[54] G. Shani, J. Pineau, and R. Kaplow, “A survey of point-based pomdp solvers,” *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.

[55] K. A. Svendsen and M. L. Seto, “Partially observable Markov decision processes for fault management in autonomous underwater vehicles,” in *2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2020, pp. 1–7. DOI: 10.1109/CCECE47787.2020.9255782.

[56] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, “Control barrier functions: Theory and applications,” in *2019 18th European Control Conference (ECC)*, 2019, pp. 3420–3431.

[57] M. Tipaldi and B. Bruenjes, “Survey on fault detection, isolation, and recovery strategies in the space domain,” *Journal of Aerospace Information Systems*, vol. 12, no. 2, pp. 235–256, 2015.

[58] M. Visinsky, J. Cavallaro, and I. Walker, “Robotic fault detection and fault tolerance: A survey,” *Reliability Engineering & System Safety*, vol. 46, no. 2, pp. 139–158, 1994, ISSN: 0951-8320.

[59] R. Mattone and A. De Luca, “Relaxed fault detection and isolation: An application to a nonlinear case study,” *Automatica*, vol. 42, no. 1, pp. 109–116, 2006.

[60] F. Baghernezhad and K. Khorasani, “Computationally intelligent strategies for robust fault detection, isolation, and identification of mobile robots,” *Neurocomputing*, vol. 171, pp. 335–346, 2016.

[61] K. Tidriri, N. Chatti, S. Verron, and T. Tiplica, "Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges," *Annual Reviews in Control*, vol. 42, pp. 63–81, 2016, ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2016.09.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578816300669.

[62] E. Khalastchi and M. Kalech, "On fault detection and diagnosis in robotic systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–24, 2018.

[63] A. Marino, F. Pierri, and F. Arrichiello, "Distributed fault detection isolation and accommodation for homogeneous networked discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4840–4847, 2017. DOI: 10.1109/TAC.2017.2694556.

[64] S. Hayden, A. Sweet, and S. Christa, "Livingstone model-based diagnosis of earth observing one," in *AIAA 1st Intelligent Systems Technical Conference*, 2004, p. 6225.

[65] R. Mackey, A. Nikora, C. Altenbuchner, *et al.*, "On-board model based fault diagnosis for cubesat attitude control subsystem: Flight data results," in *2021 IEEE Aerospace Conference*, 2021, pp. 1–17.

[66] M. Šimandl and I. Punčochář, "Active fault detection and control: Unified formulation and optimal design," *Automatica*, vol. 45, no. 9, pp. 2052–2059, 2009.

[67] M. Sampath, S. Lafortune, and D. Teneketzis, "Active diagnosis of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 7, pp. 908–929, 1998.

[68] E. Chanthery, Y. Pencolé, and N. Bussac, "An ao*-like algorithm implementation for active diagnosis," in *10th International Symposium on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS*, Citeseer, 2010, pp. 75–76.

[69] E. Chanthery, L. Travé-Massuyès, Y. Pencolé, R. De Ferluc, and B. Dellandrea, "Applying active diagnosis to space systems by on-board control procedures," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 5, pp. 2568–2580, 2019.

[70] L. Blackmore and B. Williams, "Finite horizon control design for optimal discrimination between several models," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 1147–1152.

[71] J. A. Paulson, T. A. N. Heirung, R. D. Braatz, and A. Mesbah, "Closed-loop active fault diagnosis for stochastic linear systems," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 735–741. DOI: 10.23919/ACC.2018.8431031.

[72] J. K. Scott, R. Findeisen, R. D. Braatz, and D. M. Raimondo, "Input design for guaranteed fault diagnosis using zonotopes," *Automatica*, vol. 50, no. 6, pp. 1580–1589, 2014.

[73] D. M. Raimondo, G. R. Marseglia, R. D. Braatz, and J. K. Scott, "Closed-loop input design for guaranteed fault diagnosis using set-valued observers," *Automatica*, vol. 74, pp. 107–117, 2016.

[74] S. L. Campbell and R. Nikoukhah, *Auxiliary signal design for failure detection*. Princeton University Press, 2015, vol. 11.

[75] S. Campbell, K. Horton, and R. Nikoukhah, "Auxiliary signal design for rapid multi-model identification using optimization," *Automatica*, vol. 38, no. 8, pp. 1313–1325, 2002.

[76] S. Campbell, K. Drake, I. Andjelkovic, K. Sweetingham, and D. Choe, "Model based failure detection using test signals from linearizations: A case study," in *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, 2006, pp. 2659–2664.

[77] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[78] K. Hang, W. G. Bircher, A. S. Morgan, and A. M. Dollar, "Manipulation for self-identification, and self-identification for better manipulation," *Science Robotics*, vol. 6, no. 54, eabe1321, 2021.

[79] B. Chen, R. Kwiatkowski, C. Vondrick, and H. Lipson, "Fully body visual self-modeling of robot morphologies," *Science Robotics*, vol. 7, no. 68, eabn1944, 2022.

[80] E. Altman, *Constrained Markov decision processes*. Routledge, 2021.

[81] P. Poupart, A. Malhotra, P. Pei, K.-E. Kim, B. Goh, and M. Bowling, "Approximate linear programming for constrained partially observable Markov decision processes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.

[82] D. Kim, J. Lee, K.-E. Kim, and P. Poupart, "Point-based value iteration for constrained POMDPs," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 11, 2011, pp. 1968–1974.

[83] K. H. Wray and K. Czuprynski, "Scalable gradient ascent for controllers in constrained POMDPs," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 9085–9091.

[84] A. Undurti and J. P. How, "An online algorithm for constrained POMDPs," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 3966–3973.

[85] J. Lee, G.-h. Kim, P. Poupart, and K.-E. Kim, “Monte-carlo tree search for constrained POMDPs,” in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018.

[86] A. Jamgochian, A. Corso, and M. J. Kochenderfer, “Online planning for constrained POMDPs with continuous spaces through dual ascent,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 33, 2023, pp. 198–202.

[87] S. Thiébaux, B. Williams, *et al.*, “Rao*: An algorithm for chance-constrained POMDP’s,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[88] M. Khonji, A. Jasour, and B. C. Williams, “Approximability of constant-horizon constrained POMDP.,” in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 5583–5590.

[89] S. Hong, S. U. Lee, X. Huang, M. Khonji, R. Alyassi, and B. C. Williams, “An anytime algorithm for chance constrained stochastic shortest path problems and its application to aircraft routing,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 475–481. DOI: 10.1109/ICRA48506.2021.9561229.

[90] M. P. Vitus and C. J. Tomlin, “Closed-loop belief space planning for linear, Gaussian systems,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2152–2159. DOI: 10.1109/ICRA.2011.5980257.

[91] V. Indelman, L. Carlone, and F. Dellaert, “Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments,” *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 849–882, 2015.

[92] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, “Mastering atari with discrete world models,” *arXiv preprint arXiv:2010.02193*, 2020.

[93] R. Rafailov, T. Yu, A. Rajeswaran, and C. Finn, “Offline reinforcement learning from images with latent space models,” in *Proceedings of Machine Learning Research*, 2021, pp. 1154–1168.

[94] D. Ghosh, A. Ajay, P. Agrawal, and S. Levine, “Offline RL policies should be trained to be adaptive,” in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162, Jul. 2022, pp. 7513–7530.

[95] R. He, E. Brunskill, and N. Roy, “Puma: Planning under uncertainty with macro-actions,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, 2010, pp. 1089–1095.

[96] H. Ma and J. Pineau, “Information gathering and reward exploitation of subgoals for pomdps,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.

[97] M. T. Spaan, T. S. Veiga, and P. U. Lima, “Decision-theoretic planning under uncertainty with information rewards for active cooperative perception,” *Autonomous Agents and Multi-Agent Systems*, vol. 29, pp. 1157–1185, 2015.

[98] L. Dressel and M. Kochenderfer, “Efficient decision-theoretic target localization,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 27, 2017, pp. 70–78.

[99] J. C. Saborío and J. Hertzberg, “Towards domain-independent biases for action selection in robotic task-planning under uncertainty.,” in *International Conference on Agents and Artificial Intelligence*, 2018, pp. 85–93.

[100] A. Dixit, M. Ahmadi, and J. W. Burdick, “Risk-sensitive motion planning using entropic value-at-risk,” in *2021 European Control Conference (ECC)*, 2021, pp. 1726–1732.

[101] R. H. Byrd, J. C. Gilbert, and J. Nocedal, “A trust region method based on interior point techniques for nonlinear programming,” *Mathematical programming*, vol. 89, pp. 149–185, 2000.

[102] H. Tsukamoto, B. Rivière, C. Choi, A. Rahmani, and S.-J. Chung, “CaRT: Certified safety and robust tracking in learning-based motion planning for multi-agent systems,” in *2023 62nd IEEE Conference on Decision and Control (CDC)*, 2023, pp. 2910–2917.

[103] M. Vahs, C. Pek, and J. Tumova, “Belief control barrier functions for risk-aware control,” *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 8565–8572, 2023. DOI: 10.1109/LRA.2023.3330662.

[104] Z. Laouar, R. Mazouz, T. Becker, Q. H. Ho, and Z. N. Sunberg, “Feasibility-guided safety-aware model predictive control for jump Markov linear systems,” *arXiv preprint arXiv:2310.14116*, 2023.

[105] S. C. Surace, A. Kutschireiter, and J.-P. Pfister, “How to avoid the curse of dimensionality: Scalability of particle filters with and without importance weights,” *SIAM Review*, vol. 61, no. 1, pp. 79–91, 2019.

[106] S. Basu, S. Rajesh, K. Zheng, S. Tellex, and R. I. Bahar, “Parallelizing POMCP to solve complex POMDPs,” in *Robotics: Science and Systems (RSS) Workshop on Software Tools for Real-time Optimal Control*, 2021.

[107] P. Cai, Y. Luo, D. Hsu, and W. S. Lee, “Hyp-despot: A hybrid parallel algorithm for online planning under uncertainty,” *The International Journal of Robotics Research*, vol. 40, no. 2-3, pp. 558–573, 2021.

[108] R. D. Lorenz, "Calculating risk and payoff in planetary exploration and life detection missions," *Advances in Space Research*, vol. 64, no. 4, pp. 944–956, 2019, ISSN: 0273-1177. DOI: `https://doi.org/10.1016/j.asr.2019.05.026`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0273117719303539`.

[109] G. Oriolo, G. Ulivi, and M. Vendittelli, "Real-time map building and navigation for autonomous robots in unknown environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 3, pp. 316–333, 1998. DOI: `10.1109/3477.678626`.

[110] P. Arm, G. Waibel, J. Preisig, *et al.*, "Scientific exploration of challenging planetary analog environments with a team of legged robots," *Science Robotics*, vol. 8, no. 80, eade9548, 2023.

[111] T. Lew, A. Sharma, J. Harrison, A. Bylard, and M. Pavone, "Safe active dynamics learning and control: A sequential exploration–exploitation framework," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2888–2907, 2022. DOI: `10.1109/TRO.2022.3154715`.

[112] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6059–6066. DOI: `10.1109/CDC.2018.8619572`.

[113] M. O'Connell, G. Shi, X. Shi, *et al.*, "Neural-fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, eabm6597, 2022.

[114] E. S. Lupu, F. Xie, J. A. Preiss, J. Alindogan, M. Anderson, and S.-J. Chung, "Magic vfm-meta-learning adaptation for ground interaction control with visual foundation models," *IEEE Transactions on Robotics*, 2024.

[115] K. Rocki and R. Suda, "Large-scale parallel monte carlo tree search on gpu," in *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, 2011, pp. 2034–2037. DOI: `10.1109/IPDPS.2011.370`.

[116] N. A. Barriga, M. Stanescu, and M. Buro, "Parallel uct search on gpus," in *2014 IEEE Conference on Computational Intelligence and Games*, 2014, pp. 1–7. DOI: `10.1109/CIG.2014.6932879`.

[117] M. Pharr and R. Fernando, *GPU Gems 2: Programming techniques for high-performance graphics and general-purpose computation (gpu gems)*. Addison-Wesley Professional, 2005, ch. 34.

[118] J. Bradbury, R. Frostig, P. Hawkins, *et al.*, *JAX: Composable transformations of Python+NumPy programs*, version 0.3.13, 2018.

[119] C. V. Ramamoorthy and H. F. Li, "Pipeline architecture," *ACM Comput. Surv.*, vol. 9, no. 1, pp. 61–102, Mar. 1977, ISSN: 0360-0300. DOI: `10.1145/356683.356687`. [Online]. Available: `https://doi.org/10.1145/356683.356687`.

[120] R. Murphy, "On the effects of memory latency and bandwidth on supercomputer application performance," in *2007 IEEE 10th International Symposium on Workload Characterization*, 2007, pp. 35–43. DOI: `10.1109/IISWC.2007.4362179`.

[121] S. E. Anderson, "Bit twiddling hacks," *URL: http://www.arvifox.com/wp-content/uploads/2015/08/Bit-Twiddling-Hacks.pdf*, 2005.

[122] A. Angelou, A. Dadaliaris, M. Dossis, and G. Dimitriou, "Branchless code generation for modern processor architectures," in *Proceedings of the 25th Pan-Hellenic Conference on Informatics*, ser. PCI '21, Volos, Greece: Association for Computing Machinery, 2022, pp. 300–305, ISBN: 9781450395557. DOI: `10.1145/3503823.3503879`. [Online]. Available: `https://doi.org/10.1145/3503823.3503879`.

[123] G. M. .-. Chaslot, M. H. Winands, and H. J. van Den Herik, "Parallel monte-carlo tree search," in *Computers and Games: 6th International Conference, CG 2008, Beijing, China, September 29-October 1, 2008. Proceedings 6*, Springer, 2008, pp. 60–71.

[124] T. Cazenave and N. Jouandeau, "On the Parallelization of UCT," in *Proceedings of the Computer Games Workshop*, Amsterdam, Netherlands, Jun. 2007. [Online]. Available: `https://hal.science/hal-02310186`.

[125] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[126] B. Riviere, W. Hönig, M. Anderson, and S.-J. Chung, "Neural tree expansion for multi-robot planning in non-cooperative environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6868–6875, 2021.

[127] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk, "Monte carlo tree search: A review of recent modifications and applications," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 2497–2562, 2023.

[128] J.-I. Aoe, "An efficient digital search algorithm by using a double-array structure," *IEEE Transactions on Software Engineering*, vol. 15, no. 9, pp. 1066–1077, 1989. DOI: `10.1109/32.31365`.

[129] A. Elmasry and J. Katajainen, "Branchless search programs," in *Experimental Algorithms*, V. Bonifaci, C. Demetrescu, and A. Marchetti-Spaccamela, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 127–138, ISBN: 978-3-642-38527-8.

[130] T. D. Han and T. S. Abdelrahman, "Reducing branch divergence in gpu programs," in *Proceedings of the fourth workshop on general purpose processing on graphics processing units*, 2011, pp. 1–8.

[131] P. Hijma, S. Heldens, A. Sclocco, B. van Werkhoven, and H. E. Bal, "Optimization techniques for gpu programming," *ACM Comput. Surv.*, vol. 55, no. 11, Mar. 2023, ISSN: 0360-0300. DOI: 10.1145/3570638. [Online]. Available: https://doi.org/10.1145/3570638.

[132] M. Aung, A. Ahmed, M. Wette, *et al.*, "An overview of formation flying technology development for the terrestrial planet finder mission," vol. 4, Apr. 2004, 2667–2679 Vol.4, ISBN: 0-7803-8155-6. DOI: 10.1109/AERO.2004.1368062.

[133] S. P. Hughes and F. H. Bauer, "Preliminary optimal orbit design for the laser interferometer space antenna (lisa)," in *25th Annual Guidance and Control Conference*, 2002.

[134] M. Lavalle, I. Seker, J. Ragan, *et al.*, "Distributed aperture radar tomographic sensors (darts) to map surface topography and vegetation structure," in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, 2021, pp. 1090–1093. DOI: 10.1109/IGARSS47720.2021.9553170.

[135] G. Zhai, J. Zhang, and Z. Zhou, "On-orbit target tracking and inspection by satellite formation," *Journal of Systems Engineering and Electronics*, vol. 24, no. 6, pp. 879–888, 2013. DOI: 10.1109/JSEE.2013.00102.

[136] H. Nourzadeh and J. McInroy, "Multi-agent orbit design for visual perception enhancement purpose," *International Journal of Advanced Robotic Systems*, vol. 11, no. 10, p. 161, 2014. DOI: 10.5772/58894. eprint: https://doi.org/10.5772/58894. [Online]. Available: https://doi.org/10.5772/58894.

[137] J. Lathrop, W. Cook, J. Ragan, and S.-J. Chung, "Applying monte carlo tree search for orbit selection in multi-agent inspection," in *2022 AAS/AIAA Astrodynamics Specialist Conference*, 2022.

[138] G. Falco, A. Viswanathan, and A. Santangelo, "Cubesat security attack tree analysis," in *2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, IEEE, 2021, pp. 68–76.

[139] M. Manulis, C. P. Bridges, R. Harrison, V. Sekar, and A. Davis, "Cyber security in new space: Analysis of threats, key enabling technologies and challenges," *International Journal of Information Security*, vol. 20, pp. 287–311, 2021.

[140] J. Willbold, M. Schloegel, M. Vögele, M. Gerhardt, T. Holz, and A. Abbasi, "Space odyssey: An experimental software security analysis of satellites," in *2023 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2023, pp. 1–19.

[141] K. Matsuka, A. O. Feldman, E. S. Lupu, S.-J. Chung, and F. Y. Hadaegh, "Decentralized formation pose estimation for spacecraft swarms," *Advances in Space Research*, vol. 67, no. 11, pp. 3527–3545, 2021.

[142] K. Matsuka, A. Santamaria-Navarro, V. Capuano, A. Harvard, A. Rahmani, and S.-J. Chung, "Collaborative pose estimation of an unknown target using multiple spacecraft," in *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–11.

[143] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, "Attack models and scenarios for networked control systems," in *Proceedings of the 1st international conference on High Confidence Networked Systems*, 2012, pp. 55–64.

[144] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 11, pp. 2715–2729, 2013. DOI: `10.1109/TAC.2013.2266831`.

[145] S. D. Bopardikar and A. Speranzon, "On analysis and design of stealth-resilient control systems," in *2013 6th International Symposium on Resilient Control Systems (ISRCS)*, IEEE, 2013, pp. 48–53.

[146] A. M. Teixeira, "Security metrics for control systems," in *Safety, Security and Privacy for Cyber-Physical Systems*, Springer, 2021, pp. 99–121.

[147] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, *et al.*, "Limiting the impact of stealthy attacks on industrial control systems," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, Vienna, Austria: Association for Computing Machinery, 2016, pp. 1092–1105, ISBN: 9781450341394. DOI: `10.1145/2976749.2978388`. [Online]. Available: `https://doi.org/10.1145/2976749.2978388`.

[148] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, 2015, ISSN: 0005-1098. DOI: `https://doi.org/10.1016/j.automatica.2014.10.067`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0005109814004488`.

[149] T. Sui, Y. Mo, D. Marelli, X. Sun, and M. Fu, "The vulnerability of cyber-physical system under stealthy attacks," *IEEE Transactions on Automatic Control*, vol. 66, no. 2, pp. 637–650, 2021. DOI: `10.1109/TAC.2020.2987307`.

[150] R. L. Dobrushin, "Prescribing a system of random variables by conditional distributions," *Theory of Probability & Its Applications*, vol. 15, no. 3, pp. 458–486, 1970.

[151] L. N. Vaserstein, "Markov processes over denumerable products of spaces, describing large systems of automata," *Problemy Peredachi Informatsii*, vol. 5, no. 3, pp. 64–72, 1969.

[152] D. Li and S. Martínez, "High-confidence attack detection via wasserstein-metric computations," *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 379–384, 2021. DOI: 10.1109/LCSYS.2020.3002689.

[153] V. Renganathan, N. Hashemi, J. Ruths, and T. H. Summers, "Distributionally robust tuning of anomaly detectors in cyber-physical systems with stealthy attacks," in *2020 American Control Conference (ACC)*, 2020, pp. 1247–1252. DOI: 10.23919/ACC45564.2020.9147661.

[154] T. Başar and P. Bernhard, *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media, 2008.

[155] T. Başar and G. J. Olsder, *Dynamic noncooperative game theory*. SIAM, 1998.

[156] M. Sion, "On general minimax theorems.," *Pacific Journal of Mathematics*, 1958.

[157] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-carlo tree search: A new framework for game ai," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 4, 2008, pp. 216–217.

[158] S. Gelly and D. Silver, "Achieving master level play in 9 x 9 computer go.," in *AAAI*, vol. 8, 2008, pp. 1537–1540.

[159] S. Morris and T. Ui, "Best response equivalence," *Games and Economic Behavior*, vol. 49, no. 2, pp. 260–287, 2004.

[160] H. Moulin and J. .-. Vial, "Strategically zero-sum games: The class of games whose completely mixed equilibria cannot be improved upon," *International Journal of Game Theory*, vol. 7, pp. 201–221, 1978.

[161] G. Ostrovski, "Topics arising from fictitious play dynamics," Ph.D. dissertation, University of Warwick, 2013.

[162] Z. N. Sunberg and M. J. Kochenderfer, "Online algorithms for POMDPs with continuous state, action, and observation spaces," in *ICAPS*, AAAI Press, 2018, pp. 259–263.

[163] G. F. Lawler and V. Limic, *Random walk: a modern introduction*. Cambridge University Press, 2010, vol. 123.

[164] M. Zeballos, C. S. Fumagalli, S. M. Ghelfi, and A. Schwaninger, "Why and how unpredictability is implemented in aviation security–a first qualitative study," *Heliyon*, vol. 9, no. 3, 2023.

[165] S. M. Ghelfi-Waechter, A. Bearth, C. S. Fumagalli, and F. Hofer, "Towards unpredictability in airport security," *Journal of Airport Management*, vol. 13, no. 2, pp. 110–121, 2019.

[166] U. Haldimann, “Unpredictability in aviation security: How to improve the effectiveness of current security concepts by adding the element of surprise,” *Journal of Airport Management*, vol. 12, no. 1, pp. 5–12, 2018.

[167] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, “Revealing stealthy attacks in control systems,” in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2012, pp. 1806–1813. DOI: `10.1109/Allerton.2012.6483441`.

[168] Y. Mao, H. Jafarnejadsani, P. Zhao, E. Akyol, and N. Hovakimyan, “Novel stealthy attack and defense strategies for networked control systems,” *IEEE Transactions on Automatic Control*, vol. 65, no. 9, pp. 3847–3862, 2020. DOI: `10.1109/TAC.2020.2997363`.

[169] M. Bahrami and H. Jafarnejadsani, “Detection of stealthy adversaries for networked unmanned aerial vehicles,” in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022, pp. 1111–1120. DOI: `10.1109/ICUAS54217.2022.9836208`.

[170] H. Tsukamoto, J. D. Ibrahim, J. Hajar, J. Ragan, S.-J. Chung, and F. Y. Hadaegh, “Robust optimal network topology switching for zero dynamics attacks,” in *2024 63nd IEEE Conference on Decision and Control (CDC) (© 2024 IEEE)*, 2024. [Online]. Available: `https://arxiv.org/abs/2407.18440`,

[171] C. De Persis and A. Isidori, “A geometric approach to nonlinear fault detection and isolation,” *IEEE Transactions on Automatic Control*, vol. 46, no. 6, pp. 853–865, 2001. DOI: `10.1109/9.928586`.

[172] T. Spyridopoulos, G. Karanikas, T. Tryfonas, and G. Oikonomou, “A game theoretic defence framework against dos/ddos cyber attacks,” *Computers & Security*, vol. 38, pp. 39–50, 2013.

[173] M. Zhang, Z. Zheng, and N. B. Shroff, “A game theoretic model for defending against stealthy attacks with limited resources,” in *Decision and Game Theory for Security: 6th International Conference, GameSec 2015, London, UK, November 4-5, 2015, Proceedings 6*, Springer, 2015, pp. 93–112.

[174] A. Attiah, M. Chatterjee, and C. C. Zou, “A game theoretic approach to model cyber attack and defense strategies,” in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7. DOI: `10.1109/ICC.2018.8422719`.

[175] A. T. Nguyen, A. M. Teixeira, and A. Medvedev, “A single-adversary-single-detector zero-sum game in networked control systems,” *IFAC-PapersOnLine*, vol. 55, no. 13, pp. 49–54, 2022.

[176] C. Wu, X. Li, W. Pan, J. Liu, and L. Wu, “Zero-sum game-based optimal secure control under actuator attacks,” *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3773–3780, 2021. DOI: `10.1109/TAC.2020.3029342`.

[177] M. Delpech, F. Malbet, T. Karlsson, R. Larsson, A. Léger, and J. Jorgensen, "Flight demonstration of formation flying capabilities for future missions (neat pathfinder)," *Acta Astronautica*, vol. 105, no. 1, pp. 82–94, Dec. 2014, ISSN: 0094-5765. DOI: `10.1016/j.actaastro.2014.05.027`. [Online]. Available: `http://dx.doi.org/10.1016/j.actaastro.2014.05.027`.

[178] N. R. Council, *Thriving on Our Changing Planet: A Decadal Strategy for Earth Observation from Space*. National Academic Press, 2017.

[179] A. Reigber and A. Moreira, "First demonstration of airborne SAR tomography using multibaseline L-band data," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 38, no. 5, pp. 2142–2152, Sep. 2000.

[180] A. Moreira, P. Prats-Iraola, M. Younis, G. Krieger, I. Hajnsek, and K. P. Papathanassiou, "A tutorial on synthetic aperture radar," *IEEE Geoscience and remote sensing magazine*, vol. 1, no. 1, pp. 6–43, 2013.

[181] M. B. Wooten and I. D. Walker, "A novel vine-like robot for in-orbit inspection," 45th International Conference on Environmental Systems, 2015.

[182] J. P. Davis, J. P. Mayberry, and J. P. Penn, "On-orbit servicing: Inspection repair refuel upgrade and assembly of satellites in space," *The Aerospace Corporation, report*, 2019.

[183] T. P. Setterfield, "On-orbit inspection of a rotating object using a moving observer," Ph.D. dissertation, Massachusetts Institute of Technology, 2017.

[184] J. Sullivan, S. Grimberg, and S. D'Amico, "Comprehensive survey and assessment of spacecraft relative motion dynamics models," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 8, pp. 1837–1859, 2017.

[185] W. H. CLOHESSY and R. S. WILTSHIRE, "Terminal guidance system for satellite rendezvous," *Journal of the Aerospace Sciences*, vol. 27, no. 9, pp. 653–658, 1960. DOI: `10.2514/8.8704`. eprint: `https://doi.org/10.2514/8.8704`. [Online]. Available: `https://doi.org/10.2514/8.8704`.

[186] D. Morgan, S.-J. Chung, L. Blackmore, B. Acikmese, D. Bayard, and F. Y. Hadaegh, "Swarm-keeping strategies for spacecraft under j2 and atmospheric drag perturbations," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 5, pp. 1492–1506, 2012.

[187] J. Curlander and R. McDonough, *Synthetic aperture radar: systems and signal processing* (Wiley series in remote sensing). Wiley, 1991, ISBN: 9780471857709.

[188] J. Homer, I. Longstaff, and G. Callaghan, "High resolution 3-D SAR via multi-baseline interferometry," in *Geoscience and Remote Sensing Symposium, 1996. IGARSS '96. 'Remote Sensing for a Sustainable Future.', International*, vol. 1, May 1996, 796–798 vol.1.

[189] G. Fornaro, F. Serafino, and F. Soldovieri, “Three-dimensional focusing with multipass SAR data,” *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 41, no. 3, pp. 507–517, Mar. 2003.

[190] G. Fornaro, F. Lombardini, and F. Serafino, “Three-dimensional multipass SAR focusing: Experiments with long-term spaceborne data,” *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 43, no. 4, pp. 702–714, Apr. 2005.

[191] S. Tebaldini, “Algebraic synthesis of forest scenarios from multibaseline PolInSAR data,” *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 47, no. 12, pp. 4132–4142, Dec. 2009.

[192] M. Lavalle, M. Simard, and S. Hensley, “A temporal decorrelation model for polarimetric radar interferometers,” *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 50, no. 7, pp. 2880–2888, Jul. 2012, ISSN: 0196-2892. DOI: `10.1109/TGRS.2011.2174367`.

[193] H. A. Zebker and J. Villasenor, “Decorrelation in interferometric radar echoes,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 30, no. 5, pp. 950–959, Sep. 1992.

[194] I. Seker and M. Lavalle, “Tomographic performance of multi-static radar formations: Theory and simulations,” *Remote Sensing*, vol. 13, no. 4, p. 737, 2021.

[195] Y. K. Nakka, W. Hönig, C. Choi, A. Harvard, A. Rahmani, and S.-J. Chung, “Information-based guidance and control architecture for multi-spacecraft on-orbit inspection,” *AIAA Scitech 2021 Forum*, 2021.

[196] M. Schwager, D. Rus, and J.-J. Slotine, “Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment,” *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 371–383, 2011.

[197] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 1999.

[198] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, 1970.

[199] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Publishing Co, 1989.

[200] S. Spiridonova and R. Kahle, “Hrws - an ambitious 4+ satellite formation flying mission,” in *27th International Symposium on Space Flight Dynamics*, 2019. [Online]. Available: `https://elib.dlr.de/127795/`.

[201] G. Krieger, M. Zink, M. Bachmann, *et al.*, “Tandem-x: A radar interferometer with two formation-flying satellites,” *Acta Astronautica*, vol. 89, pp. 83–98, 2013, ISSN: 0094-5765. DOI: `https://doi.org/10.1016/j.actaastro.2013.03.008`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0094576513000805`.

[202] G. Krieger, I. Hajnsek, K. P. Papathanassiou, M. Younis, and A. Moreira, "Interferometric synthetic aperture radar (sar) missions employing formation flying," *Proceedings of the IEEE*, vol. 98, no. 5, pp. 816–843, 2010. DOI: 10.1109/JPROC.2009.2038948.

[203] P. Lopez Dekker, H. Rott, B. Chapron, and P. Prats-Iraola, "Stereo thermo-optically enhanced radar for earth, ocean, ice, and land dynamics (stereoid)," Mar. 2018. DOI: 10.13140/RG.2.2.25804.46728.

[204] H. Rott, P. López-Dekker, S. Solberg, *et al.*, "Sesame: A single-pass interferometric sentinel-1 companion sar mission for monitoring geo- and biosphere dynamics," in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017, pp. 107–110. DOI: 10.1109/IGARSS.2017.8126905.

[205] G. Krieger, M. Zonno, M. Rodriguez-Cassola, *et al.*, "Mirrorsar: A fractionated space radar for bistatic, multistatic and high-resolution wide-swath sar imaging," in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017, pp. 149–152. DOI: 10.1109/IGARSS.2017.8126916.

[206] D.-Y. Kim, B. Woo, S.-Y. Park, and K.-H. Choi, "Hybrid optimization for multiple-impulse reconfiguration trajectories of satellite formation flying," *Advances in Space Research*, vol. 44, no. 11, pp. 1257–1269, 2009, ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2009.07.029. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0273117709005390.

[207] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model predictive control of swarms of spacecraft using sequential convex programming," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 6, pp. 1725–1740, 2014.

[208] R. Foust, S.-J. Chung, and F. Y. Hadaegh, "Optimal guidance and control with nonlinear dynamics using sequential convex programming," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 4, pp. 633–644, 2020.

[209] Y. K. K. Nakka, W. Hönig, C. Choi, A. Harvard, A. Rahmani, and S.-J. Chung, "Information-based guidance and control architecture for multi-spacecraft on-orbit inspection," in *AIAA Scitech 2021 Forum*. DOI: 10.2514/6.2021-1103. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2021-1103. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2021-1103.

[210] P. Palmer, "Optimal relocation of satellites flying in near-circular-orbit formations," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 3, pp. 519–526, 2006. DOI: 10.2514/1.14310. eprint: https://doi.org/10.2514/1.14310. [Online]. Available: https://doi.org/10.2514/1.14310.

[211] C. W. T. Roscoe, S. R. Vadali, K. T. Alfriend, and U. P. Desai, "Optimal formation design for magnetospheric multiscale mission using differential orbital elements," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 1070–1080, 2011. DOI: `10.2514/1.52484`. eprint: `https://doi.org/10.2514/1.52484`. [Online]. Available: `https://doi.org/10.2514/1.52484`.

[212] A. W. Koenig, S. D'Amico, B. Macintosh, and C. J. Titus, "Optimal formation design of a miniaturized distributed occulter/telescope in earth orbit," in *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference*, 2015.

[213] B. Bernhard, C. Choi, A. Rahmani, S.-J. Chung, and F. Hadaegh, "Coordinated motion planning for on-orbit satellite inspection using a swarm of small-spacecraft," in *2020 IEEE Aerospace Conference*, 2020, pp. 1–13. DOI: `10.1109/AERO47225.2020.9172747`.

[214] K. Yun, C. Choi, R. Alimo, *et al.*, "Multi-agent motion planning using deep learning for space applications," in *ASCEND 2020*. DOI: `10.2514/6.2020-4233`. eprint: `https://arc.aiaa.org/doi/pdf/10.2514/6.2020-4233`. [Online]. Available: `https://arc.aiaa.org/doi/abs/10.2514/6.2020-4233`.

[215] S. Glavaski and M. Elgersma, "Active aircraft fault detection and isolation," in *2001 IEEE Autotestcon Proceedings. IEEE Systems Readiness Technology Conference. (Cat. No.01CH37237)*, 2001, pp. 692–705. DOI: `10.1109/AUTEST.2001.949453`.

[216] R. Munos and A. W. Moore, "Variable resolution discretization in optimal control," *Machine Learning*, vol. 49, no. 2-3, pp. 291–323, 2002.

[217] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.

[218] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[219] M. Lofqvist and J. Cano, "Accelerating deep learning applications in space," *CoRR*, vol. abs/2007.11089, 2020.

[220] G. Xu and D. Wang, "Nonlinear dynamic equations of satellite relative motion around an oblate earth," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 5, pp. 1521–1524, 2008. DOI: `10.2514/1.33616`. eprint: `https://doi.org/10.2514/1.33616`. [Online]. Available: `https://doi.org/10.2514/1.33616`.

Appendix A

FEAST IMPLEMENTATION DETAILS

Shared System Parameters

The default parameters that were used in each experiment were as follows. The actuator influence is 0.1 m/s (1 DOF single integrator system) or 0.1 N (2 DOF double integrator system) with a spacecraft mass of 1 kg. The action space consists of combinations of up to 3 thrusters firing at a constant value. This space is further restricted by requiring the chosen action to have a non-zero acceleration on the nominal system. In a fashion, this is the opposite approach of some traditional active fault estimation methods, which sought to inject signals in the null space of a vehicle so that it would be unaffected unless a fault had occurred [215]. The system noise is set as a single parameter, σ, which was varied by experiment. The discretization scheme for the particle filter used by POMCP is a zeroth order discretization [216] of $dx = 1$ m for the 1 and 2 DOF systems and $dv = 1$ m/s for the 2 DOF system. The time horizon is set to $K = 20$, the tree exploration/exploitation parameter is $c = 1.2$, and the discount factor is $\gamma = 0.9$. The simulation time step is chosen as $\Delta_t = 1.0$ s. The initial position covariance is $\sigma_0 = 0.001$. Finally the success threshold is set to 0.81, roughly corresponding to a confidence of 90%. Each experiment was performed over 1000 different random seeds, and the results averaged and summarized below.

1 DOF Single Integrator System

We specify the following system (Eq. (2.14)), sensors (Eq. (2.15)) and faults (Eq. (2.17)):

$$\begin{gathered} A = \begin{bmatrix} 1 \end{bmatrix}, \quad B = \begin{bmatrix} -\Delta_t & -\Delta_t & \Delta_t & \Delta_t \end{bmatrix}, \quad C = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \\ \Sigma_w = \sigma^2 I_2, \quad \Sigma_v = \sigma^2 I_1, \quad \phi_B \in \mathbb{B}^4, \quad \phi_C \in \mathbb{B}^2 \end{gathered} \tag{A.1}$$

where $f(x_{k-1})$ in Eq. (2.14) is now the linear relation Ax_{k-1}. The system has four actuators that affect the position of the system (two in each direction), and two sensors that sense the position. Total binary failure can affect any of these components.

2 DOF Double Integrator System

The 2 DOF system is defined with dynamics (Eq. (2.14)), sensors (Eq. (2.15)) and faults (Eq. (2.17)) as:

$$A = \begin{bmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\Delta_t & -\Delta_t & \Delta_t & \Delta_t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\Delta_t & -\Delta_t & \Delta_t & \Delta_t \end{bmatrix},$$
$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \Sigma_w = \sigma I_4, \quad \Sigma_v = \sigma I_4, \quad \phi_B \in \mathbb{B}^8, \quad \phi_C \in \mathbb{B}^4 \tag{A.2}$$

The systems has eight actuators that control the acceleration of the system, subject disturbances in each axis, and four sensors that sense the position subject to stochastic noise. Up to three simultaneous failures can affect both the sensors and actuators.

3 DOF Planar Spacecraft System

We specify the following planar satellite system where the spacecraft can translate in two dimensions and has third rotational degree of freedom modeled after the M-STAR spacecraft simulator hardware shown in Fig. 2.6. The planar spacecraft system (Eq. (2.14)), sensors (Eq. (2.15)) and faults (Eq. (2.17)):

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta_t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta_t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\Delta_t c_\theta & -\Delta_t c_\theta & \Delta_t c_\theta & \Delta_t c_\theta & \Delta_t s_\theta & \Delta_t s_\theta & -\Delta_t s_\theta & -\Delta_t s_\theta & 0 & 0 \\ -\Delta_t s_\theta & -\Delta_t s_\theta & \Delta_t s_\theta & \Delta_t s_\theta & -\Delta_t c_\theta & -\Delta_t c_\theta & \Delta_t c_\theta & \Delta_t c_\theta & 0 & 0 \\ \Delta_t \ell & -\Delta_t \ell & \Delta_t \ell & -\Delta_t \ell & \Delta_t \ell & -\Delta_t \ell & \Delta_t \ell & -\Delta_t \ell & \Delta_t & \Delta_t \end{bmatrix},$$

$$\Sigma_v = \sigma I_6, \quad \Sigma_w = \sigma I_6, \quad \phi_B \in \mathbb{B}^{10}, \quad \phi_C \in \mathbb{B}^6 \tag{A.3}$$

where θ the orientation of the spacecraft and s_θ and c_θ are shorthand for $\sin(\theta)$ and $\cos(\theta)$, respectively, making the system non-linear. The systems has eight thrusters and two reaction wheels that control the acceleration of the system, and six sensors that sense the position and orientation. In each experiment, the thrusters exert a force of 1 N, with a spacecraft mass of 1 kg. In this 3 DOF setting, each thruster also has a torque of ±.4 Nm and the reaction wheels have a torque of .05 Nm with a spacecraft moment of inertia of 4 kg/m^2. As before, any of these sensors and actuators can fail, and we allow up to three simultaneous failures.

Because of the non-linearity in this system, the linear propagation of Eq. (2.14) is only approximate. To address this, in our simulations we perform an RK45 integration scheme for the state transition between time steps, subject to the same additive noise term as in Eq. (A.3) above at each time step. We adapt the equivalent POMDP in Lemma 1 accordingly, and note that none of our analysis in Theorem 1 explicitly depend on the dynamics model, so we retain all of our theoretical guarantees of convergence and optimality.

Appendix B

S-FEAST IMPLEMENTATION DETAILS

This appendix provides implementation details for each of the quantitative experiments performed to validate s-FEAST.

Our source code is available at: `https://github.com/treyra/s-FEAST`.

Shared Experimental Parameters

We summarize the shared elements of each experiment here.

Failures:

- In the binary fault scenarios, faults with at most 3 simultaneous actuator and sensor failures are considered.
- In the continuous degradation and bias scenarios, no limit on the number of simultaneous faults was imposed, but two conditions where imposed when randomly generating faults. First, each degradation or bias generated had a 50% chance to nominal (no degradation or bias). Off nominal degradations or biases were then selected uniformly between 0 (no degradation or bias) and 1 (full degradation or bias). Second, when generating the possible failures, each generated bias (including the true bias) was repeated 5 times with different degradations, to create ambiguity between possible faults.
- The size of the failure space, N_Φ, is at most 40 randomly selected possibilities, including the true fault.
- In the binary fault scenarios, a selected failure scenario must have at least one sensor for each degree of freedom, or it is replaced to ensure observability (no restrictions are placed on actuator failures).

Actions (for s-FEAST, random, and greedy policies):

- The actions considered are combinations of up to 3 simultaneous thrusters firing.

- All thrusters fire at the same constant value.
- The action set is a random sub set of 20 different actions and consistent across experiment trials and time steps.

Initial Belief:

- Uniform prior over the failure space (note our method will also work with any initial failure belief that does not preemptively rule out the underlying failure).
- The physical belief is centered at the true initial value with a diagonal covariance matrix of $\sigma_0 = 0.001$.

Experiment Duration:

- In each experiment, we consider a duration of 15 time steps.
- This duration is selected to validate that s-FEAST remains safe over at least the planning horizon after identifying the underlying fault. Note that in all experiments, each s-FEAST algorithm achieves a 90% or higher diagnostic reward by the 10th time step.

We use the following parameters for s-FEAST in each experiment. The discretization scheme for position observations within each tree search is a zeroth order discretization [216] of $dx = 0.125$ m or 0.125 rad. The planning horizon is set to $K = 4$, the tree exploration/exploitation parameter is $c = 1.2$, and the discount factor is $\gamma = 1$ (in our hardware experiments a value of $\gamma = .9$ was used to promote immediate action). The simulation time step is chosen as $\Delta_t = 1.0$ s. The desired safety probability at each time step α is set to 0.9 in all trials. For the hardware experiments running in real-time, a nominal experiment time step of 1.3 seconds was used. For each control loop the actual wall-clock-time between action commands was used for propagating the marginalized filter.

3DOF Planar Spacecraft System

For each numerical experiment validating s-FEAST, we consider the 3 degree of freedom (3 DOF) planar satellite system defined by Eq. (A.3) in Appendix A, with the following changes to the process noise and faults:

$$\Sigma_w = \begin{bmatrix} \sigma_{w,x}^2 \frac{(\Delta t)^3}{3} & \sigma_{w,x}^2 \frac{(\Delta t)^2}{2} & 0 & 0 & 0 & 0 \\ \sigma_{w,x}^2 \frac{(\Delta t)^2}{2} & \sigma_{w,x}^2 \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{w,y}^2 \frac{(\Delta t)^3}{3} & \sigma_{w,y}^2 \frac{(\Delta t)^2}{2} & 0 & 0 \\ 0 & 0 & \sigma_{w,y}^2 \frac{(\Delta t)^2}{2} & \sigma_{w,y}^2 \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{w,\theta}^2 \frac{(\Delta t)^2}{2} & \sigma_{w,\theta}^2 \Delta t \\ 0 & 0 & 0 & 0 & \sigma_{w,\theta}^2 \frac{(\Delta t)^3}{3} & \sigma_{w,\theta}^2 \frac{(\Delta t)^2}{2} \end{bmatrix}, \tag{B.1}$$

$$\Sigma_v = \sigma_v^2 I_6, \quad \phi_B \in \mathbb{B}^{10} \text{ or } [0,1]^{20}, \quad \phi_C \in \mathbb{B}^6 \text{ or } [0,1]^{12}$$

Now we consider the actuators and sensors subject to both binary faults as well as continuous degradation and bias faults, following Eqs. (3.1) and (3.2).

The measurement noise is set as $\sigma_v = .4$, and the dynamics noise consists of acceleration disturbances in each axis of $\sigma_{w,x} = .2$, $\sigma_{w,y} = .2$, and $\sigma_{w,\theta} = .01$. These are chosen so that orientation is dependent on the selected actuations, making reorienting the spacecraft to use redundant thrusters a behavior that can be exploited by each policy. The noise model we use is suggested by Bar-Shalom *et al.* [217] as a method to capture continuous Gaussian white noise in systems evaluated at discrete time steps.

When safety chance constraints are applied to the system, we require a 90% probability of remaining safe at each time step.

As in Appendix A, we note the linear propagation of Eq. (3.1) is only approximate and perform an RK45 integration scheme for the state transition between time steps, now subject to the same additive noise term as in Eq. (B.1) above at each time step. We now adapt Lemma 3 accordingly, and note that Theorems 2-5 do not explicitly depend on the dynamics model, so we again retain all of our theoretical guarantees of convergence, optimality, and safety.

Baseline details

Control Barrier Functions (CBFs) are a control theoretic way to approach safety [56]. As in our formulation, a safe set is defined as the super-level set of a function $X_h = \{x \mid h(x) \geq 0\}$, and safety of the system can be formalized as the forward invariance of this set. Here, the function h is a CBF if it certifies the forward invariance of the safety set by satisfying: $\forall x, \exists u$ s.t. $\dot{h}(x, u) \geq -k(h(x))$, where k is a class $\mathcal{K}$ function. This has been extended to discrete time systems as D-CBFs in [48] by applying the equivalent condition (reformulated by [52]): $\forall x, \exists u$ s.t. $h(F(x, u)) \geq \beta h(x)$, where F is the state transition function. In our formulation, we take $\beta = 0$ as the least restrictive condition.

SCP is a method to solve non-convex optimal control problems in an iterative fashion. It works by convexifying the constraints at each time step about the trajectory produced by the previous iteration and optimizing against these approximations [49], [50]. With a suitable initial guess, SCP can solve a broad class of problems. We initialize SCP with an initial guess of an unforced trajectory at each time step. This guess is safe for the initial condition of each scenario considered.

As both CBF and SCP methods assume deterministic dynamics, the most likely fault state and corresponding physical state estimate are assumed when selecting an action, and a safety buffer is used to account for the process noise at each time step. A value of $1.28\lambda_{\max}(\Sigma_w)$ was adopted, as positive deviations greater or equal to 1.28 standard deviations have 10% cumulative probability for Gaussian distributions. Using obstacles with this safety buffer, if an initial state x_{t-1} was fully observable and the dynamics noise was the only uncertainty, then an action a_t that resulted in $h(\mathbb{E}[x_t]) \geq 0$ would achieve the desired safety probability of 90%.

The random and greedy policies are implemented using the same action space as s-FEAST. The CBF policy is implemented as a minimal control problem subject to the CBF constraint $h(x_{k+1}) > 1.28\sigma_w$ following the safety buffer. Since CBF theory assumes unconstrained control, we allow actions up to 20 times larger than the actuation limits of FEAST, and the problem is solved using the non-linear minimization solver built into SciPy [218]. If the solver reaches its iteration limit (for example, if the assumed system dynamics in the previous time step were inaccurate, leading to an unexpected collision and in-feasibility), the last considered action is returned.

The SCP policy is implemented at each time step as a convex program following [49], [50] to convexify the safety constraints. The policy selects form a continuous control space with the same actuation limits as s-FEAST and an initial guess of no control action. If the program is infeasible, it tries to re-solve by minimizing the safety violations. If the program is still infeasible, it returns the control input from the previous iteration or the initial guess.

All baselines employ our marginalized filter to perform belief updates between planning steps using the realized observation. This is done to isolate the performance between the baselines and s-FEAST to the quality of actions selected.

Appendix C

S-FEAST ADDITIONAL RESULTS

In this appendix we provide supplementary results to our s-FEAST analysis for the interested reader, including complete time series images of our hardware experiments, real-time performance analysis, and additional numerical simulations.

All experimental data and code needed to produce the plots presented here and Chapter 3 can be found at: `https://doi.org/10.5061/dryad.xgxd254r1`.

C.1 Additional Robotic Spacecraft Simulator Hardware Experimental Validation of s-FEAST

We present the evolution over time of a Discrete Control Barrier Function (D-CBF), FEAST, and s-FEAST on our robotic spacecraft simulator in our safety-critical scenario to demonstrate the need for our complete s-FEAST algorithm. All experiments start with the spacecraft on a crash course with respect to the model comet, with both retro thrusters completely failed. See the hardware results section of Chapter 3 for experiment details. A video summary of these experiments can be found at `https://www.youtube.com/watch?v=aJ04dlgaP0o` and an a side by side comparison at `https://youtu.be/z7Odjd4Ae_M`.

D-CBF Spacecraft Simulator Experiment

For this baseline experiment, control actions are selected at each time step by a D-CBF method to enforce the safety constraint. The D-CBF is implemented as described in baseline details subsection of Appendix B using our marginalized filter to update the failure belief between action selections, with the real-time modifications described in Chapter 3. In this experiment, the D-CBF method fails to prevent collision and the fault estimate diverges.

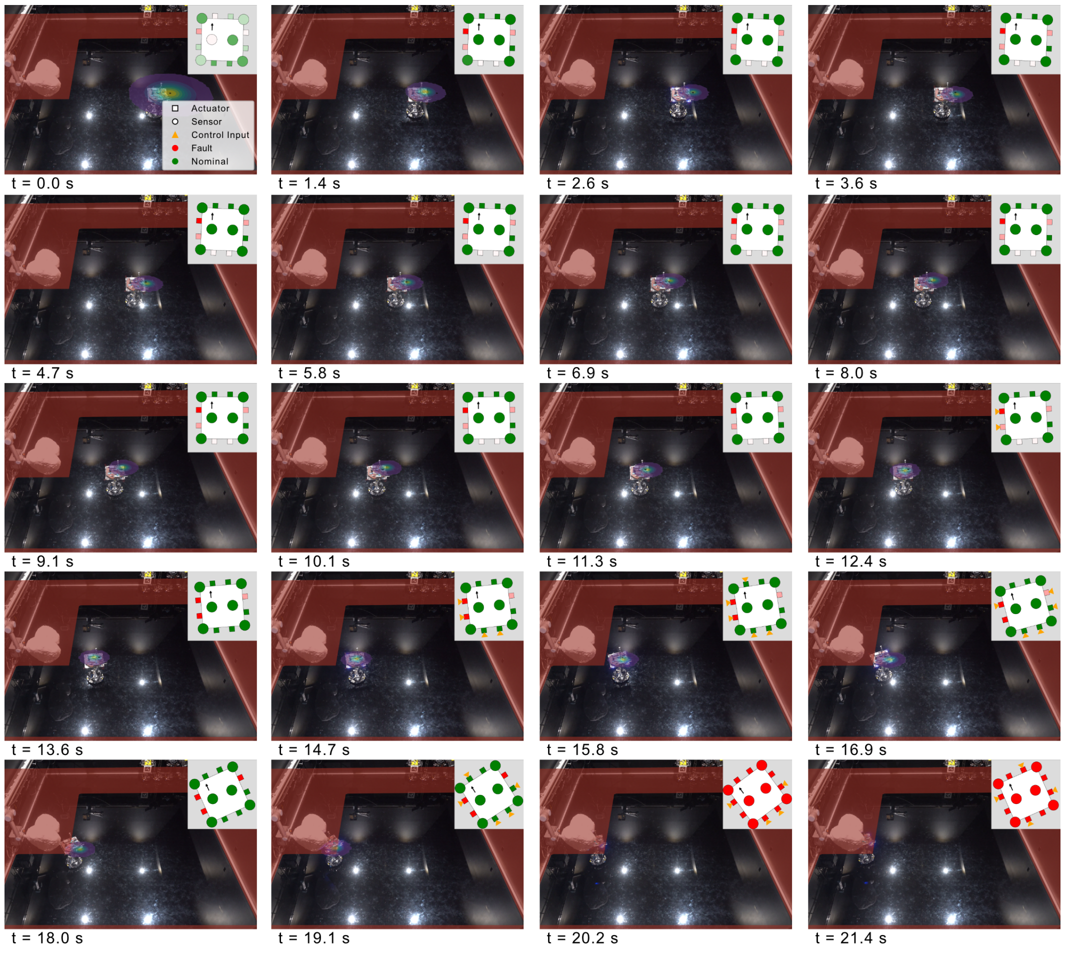


Figure C.1: **D-CBF spacecraft simulator hardware experiment.** The state and failure belief of a D-CBF method at each time step of our hardware experiment. The unsafe region is overlaid in red, the position belief overlaid as a probability density, the tree search overlaid in gray, and the failure belief overlaid in the upper right corner. The constraint, position belief, and tree overlays are approximate.

FEAST Spacecraft Simulator Experiment

We next run FEAST alone on the robotic spacecraft simulator, to demonstrate that without also considering safety, we will still crash.

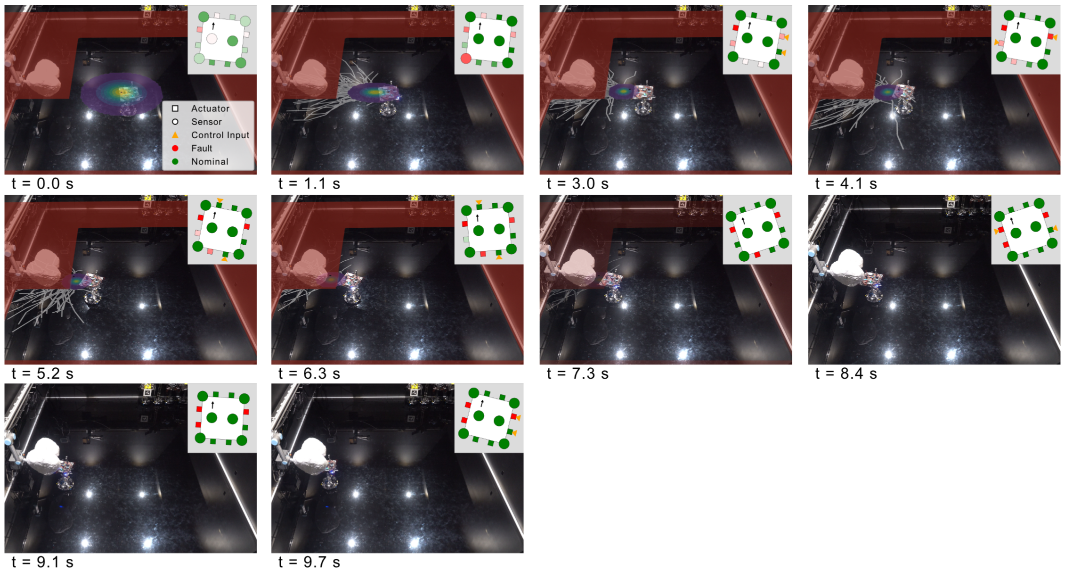


Figure C.2: **FEAST spacecraft simulator hardware experiment.** The state and failure belief of FEAST at each time step of our hardware experiment. The unsafe region is overlaid in red, the position belief overlaid as a probability density, the tree search overlaid in gray, and the failure belief overlaid in the upper right corner. The constraint, position belief, and tree overlays are approximate.

s-FEAST Spacecraft Simulator Experiment

Finally, we run s-FEAST on the robotic spacecraft simulator, to validate our methods ability to diagnose the underlying fault and remain safe.

C.2 Real-Time Performance Analysis

In this section, we discuss the implementation details of our algorithm needed to achieve the real-time performance of our hardware experiments. Our algorithm is currently implemented in Python using Google's JAX to accelerate the NumPy computations [118], primarily to accelerate the estimator propagation step, which is our main computational bottle neck. For example, a tree of 100 simulations a second, depth 4 and 40 possible failures, will compute 16,000 estimator updates a second, and we found running these large, parallelizable array operations in JAX resulted in approximately 50 times faster execution.

All the experiments we present in our work are run using JAX in the CPU only configuration to speed up our estimator prediction and update steps. Because JAX's compiled Python code cannot have any logical branching, we do not currently use JAX to accelerate the tree search itself; and running only the estimator updates on a GPU and returning to the CPU to select the next action in the tree search roll out

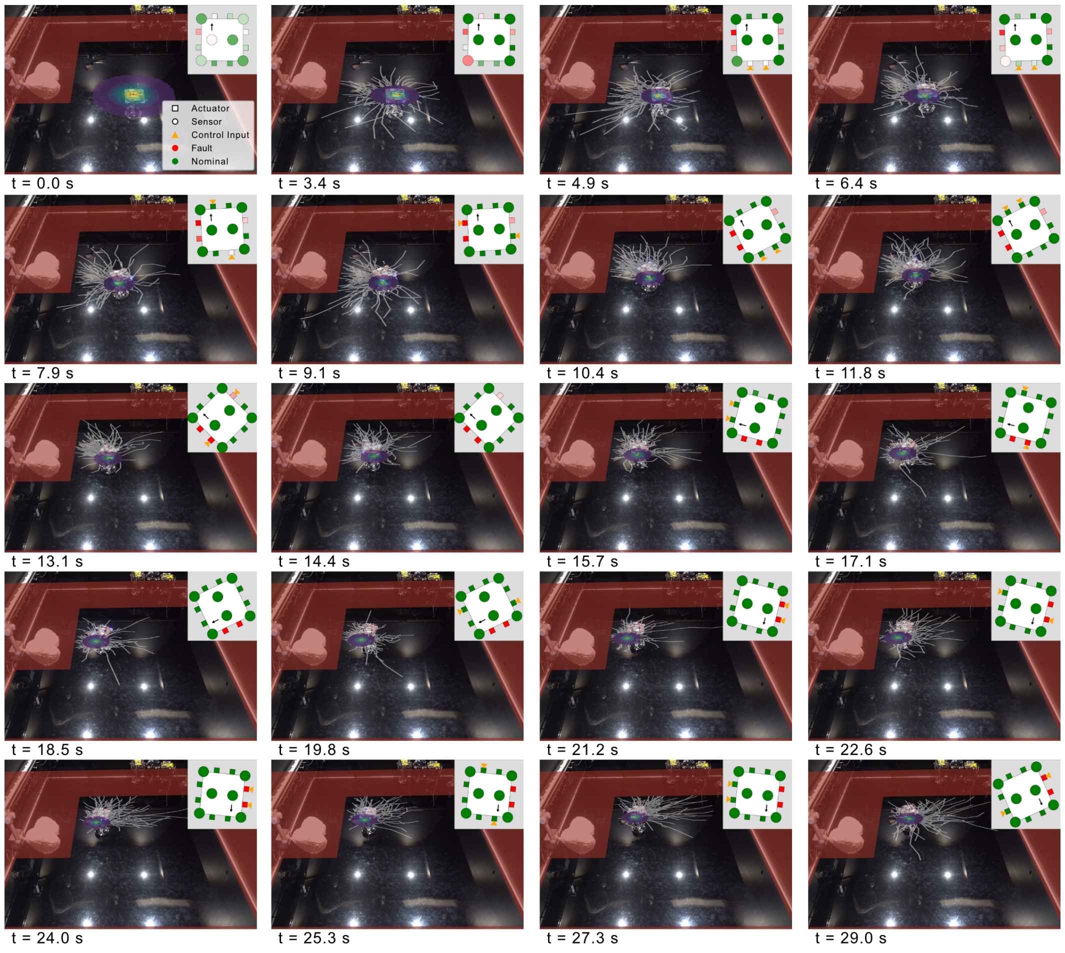


Figure C.3: **s-FEAST spacecraft simulator hardware experiment.** The state and failure belief of s-FEAST at each time step of our hardware experiment. The unsafe region is overlaid in red, the position belief overlaid as a probability density, the tree search overlaid in gray, and the failure belief overlaid in the upper right corner. The constraint, position belief, and tree overlays are approximate.

was slower in our implementation, due to the overhead of moving data on and off the GPU.

To validate s-FEAST for real-time computability, we run s-FEAST on the collision course under the adversarial binary failure scenario presented in Chapter 3, for 20 random initial conditions, and present the average wall-clock-time in Fig. C.4. We note the addition of the safety filter increases the variance of the average wall-clock-time. However, the mean value of s-FEAST's average wall-clock-time lies within one standard deviation of mean value observed when running our algorithm without the safety filter, labeled as FEAST in Fig. C.4. This validates our claims the safety filter is computationally efficient and has a negligible effect on the performance of our algorithm. Running on the robotic spacecraft, 85 simulations per time step was a typical value for the computational budget of 0.78 seconds. Since this budget does not include overhead from the logging and sensing portions of the control loop, this performance appears to match well with these experiments. These experiments are run on the NVIDIA Jetson AGX Orin 32GB Developer Kit, configured at max power settings and clock speed. The Jetson Orin modules are a family of state-of-the-art autonomy computers designed for energy-efficient robotics applications and they are considered a promising candidate platform for future space autonomy [219].

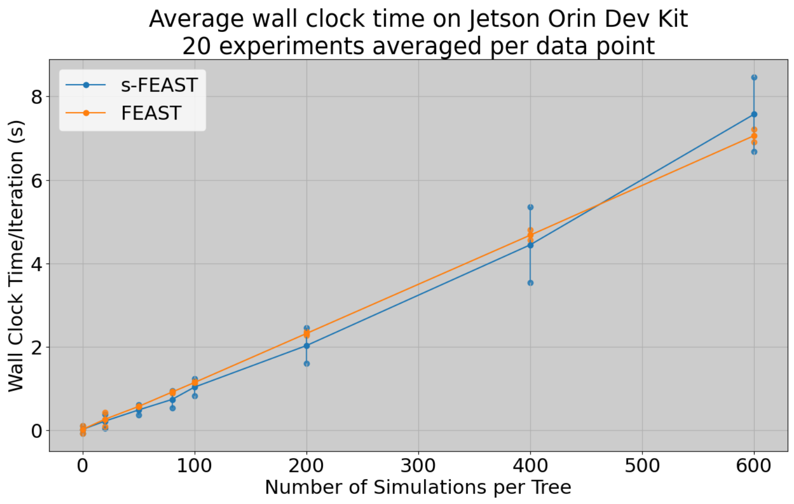


Figure C.4: **s-FEAST: Wall Clock Time.** Average computation times for s-FEAST running on Jetson Orin at increasing numbers of simulations. 1 SD error region shown for the 20 trials.

C.3 Additional Numerical Simulations Scenario: Collision Course Under Random Failures

In addition to the case of adversarial failures affecting a spacecraft on a collision course with an obstacle, we can also consider the case of random failures affecting the spacecraft. We consider both random binary failures in Fig. C.3A, and random continuous degradations and biases in Fig. C.3B. Each experiment is initialized the same as in the adversarial failure scenarios in Chapter 3, starting 10 m from the obstacle on a 1 m/s approach velocity.

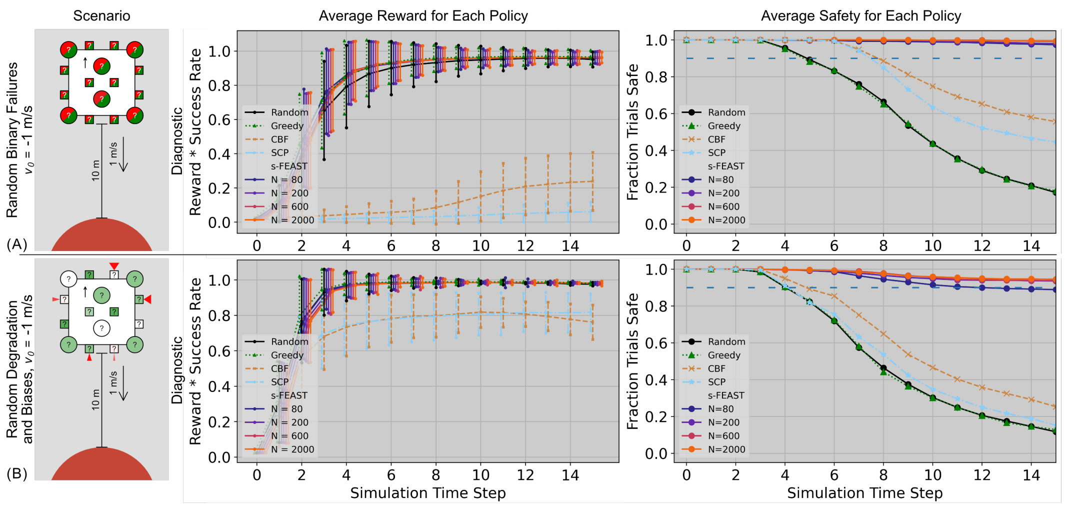


Figure C.5: **Additional validation of s-FEAST.** The numerical performance of our algorithm compared with baselines on a crash course subject to random failures. In both experiments, the robotic spacecraft starts 10 m from the obstacle with n initial 1 m/s velocity towards the obstacle.(**A**) The robot is subject to random binary failures of up to 3 components. (**B**) each component can now be randomly subjected to continuous degradation or bias, with nominal components more likely. In all experiments, s-FEAST considers 40 possible binary or general faults and starts with a uniform prior over all possibilities. In the visualization of the spacecraft component health in the left column, squares represent the thrusters and circles abstractly represent the position and orientation sensors. Green components are healthy, red are failed, and red actuations represent bias thrust of varying degrees (sensor bias is not visualized). Note that for readability, the data for each time step is artificially spread out horizontally.

Appendix D

S3AM IMPLEMENTATION DETAILS

Each of our experiments considered the angles only relative navigation problem. The formation starts at rest, with each agent in its nominal position. Estimates are updated and attacks and defenses switched after time steps of 1 second. The attacker and defender share the same initial belief centered about these nominal positions with a small covariance. These beliefs are used to select the initial attack and defense, then are independently updated using the information available to each player.

Tree search parameters:

- $M = 20$ samples were used to evaluate the stealthiness of each attack encountered in the tree.
- $K = 10$ was the max depth of each tree search.
- An exploration parameter of $c = 1.2$ was used.
- A discretization level of 1 m was used.

Experiment parameters:

- Identity matrices were used as the LQR weights for Q_k and R_k at all time steps.
- The attack set used was as follows:

$$\begin{bmatrix}0 & 0\end{bmatrix}$$
$$\begin{bmatrix}1 & 1\end{bmatrix}$$
$$\begin{bmatrix}1 & -1\end{bmatrix}$$
$$\begin{bmatrix}-1 & 1\end{bmatrix}$$
$$\begin{bmatrix}-1 & -1\end{bmatrix}$$

- The defense topology set used was as follows (recall 7 refers to the reference target, and that $d_k \in [1, \ldots, P+1]^P$ with $d_k^j \neq j$ the target of the j-th spacecraft):

$$\begin{bmatrix} 2 & 3 & 6 & 1 & 4 & 5 \end{bmatrix}$$
$$\begin{bmatrix} 4 & 1 & 2 & 5 & 6 & 3 \end{bmatrix}$$
$$\begin{bmatrix} 7 & 5 & 7 & 1 & 7 & 3 \end{bmatrix}$$
$$\begin{bmatrix} 4 & 7 & 6 & 7 & 2 & 7 \end{bmatrix}$$
$$\begin{bmatrix} 4 & 5 & 6 & 1 & 2 & 3 \end{bmatrix}$$
$$\begin{bmatrix} 2 & 1 & 7 & 5 & 4 & 7 \end{bmatrix}$$
$$\begin{bmatrix} 7 & 3 & 2 & 7 & 6 & 5 \end{bmatrix}$$
$$\begin{bmatrix} 4 & 7 & 6 & 1 & 7 & 3 \end{bmatrix}$$

- As Gaussian process and sensor noise was used, an Extended Kalman Filter was used as the estimator in each experiment. We used the same noise model as in Appendix B to capture continuous Gaussian white noise in systems evaluated at discrete time steps [217]:

$$\Sigma_w^j = \begin{bmatrix} \sigma_w^2 \frac{(\Delta t)^3}{3} & \sigma_w^2 \frac{(\Delta t)^2}{2} & 0 & 0 \\ \sigma_w^2 \frac{(\Delta t)^2}{2} & \sigma_w^2 \Delta t & 0 & 0 \\ 0 & 0 & \sigma_w^2 \frac{(\Delta t)^3}{3} & \sigma_w^2 \frac{(\Delta t)^2}{2} \\ 0 & 0 & \sigma_w^2 \frac{(\Delta t)^2}{2} & \sigma_w^2 \Delta t \frac{(\Delta t)^2}{2} \end{bmatrix}, \tag{D.1}$$

  with $\sigma_w = .01$. The same block process noise was used for each agent, including the adversarial agent.

- The sensing noise consisted of an independent one dimensional Gaussian noise with standard deviation of $\sigma_v = .01$ applied to each measurement. Note that when no range measurement was returned (due to the agent not looking at the reference target), no noise was provided.

Appendix E

ORBIT OPTIMIZATION DETAILS

In this appendix, we provide additional details on our extensions to optimal orbit designs.

E.1 Additional LVLH Frame Details

As defined in Chapter 7, the local vertical, local horizontal (LVLH) frame is given by the $\hat{x}$ (radial) direction, which is radially outward from the chief; the $\hat{z}$ (cross track) direction which is along the angular momentum vector of the chief's orbit; and the $\hat{y}$ (along track) direction which completes the right-handed coordinate system. This is in contrast to the fixed Earth-Centered Inertial (ECI) coordinate system, which is referenced from the center of an inertially fixed Earth. In this coordinate frame, $\hat{X}$ is in the direction of the vernal equinox, $\hat{Z}$ is aligned with Earth's axis, and $\hat{Y}$ completes a right handed coordinate system. The LVLH frame rotates in time with the orbiting formation relative to the ECI frame. To find the LVLH frame for a given time (for a circular orbit), the ECI frame can be transformed by a rotation about the $\hat{Z}$ axis of Ω, the right ascension of the ascending node, followed by a rotation about the transformed $\hat{x}$ axis by i, the inclination, and finally a rotation about the transformed $\hat{z}$ axis of θ, the argument of latitude at the given time instance. These coordinate systems are visualized in Fig. E.1.

While useful in locating the LVLH frame, the traditional Keplerian orbital elements are not as useful when describing the motion of a formation under perturbations. To better describe the orbit of the formation, hybrid orbital elements consisting of radial distance r, radial velocity v_x, angular momentum h, right ascension of the ascending node Ω, inclination i and argument of latitude θ are used, summarized in an orbital element vector as $[r, v_x, h, \Omega, i, \theta]$. These 6 elements are used to represent the chief orbit in the ECI frame, as the orbital dynamics under J_2 perturbations can be cleanly described using these elements [220]. From this representation the classical orbital elements can be easily recovered, and the relative motion in the LVLH frame converted back to ECI coordinates if desired.

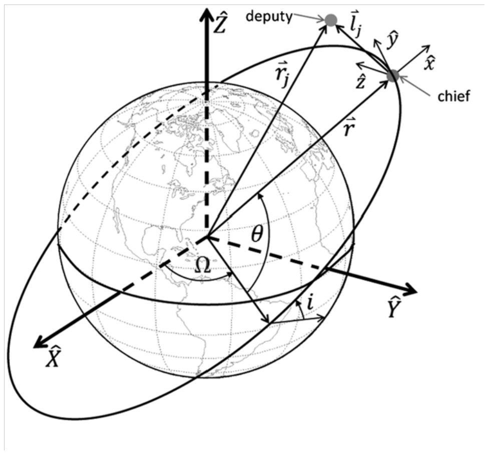


Figure E.1: **LVLH coordinate frame visualization.** The local and ECI coordinate frame and relevant orbital elements are shown for a single deputy/chief pair. Reproduced with permission from Morgan et al. [186].

E.2 Offline Orbit Design for SAR Implementation Details

Here we provide additional details about the orbital design for the DARTS mission. First, in order to solve the relative orbital mechanics, a chief orbit was selected. The NASA-ISRO Synthetic Aperture Radar (NISAR) mission was used as a reference for the orbital design, due to the similar mission goals to the DARTS mission concept. NISAR will orbit at an altitude of 747 km and an inclination of 98.4°, making it a periodic sun synchronous orbit with a repeat frequency of 12 days, during which it will complete 173 orbits.

A periodic sun synchronous orbit leverages the perturbation on the right ascension of the ascending node due to J_2, which causes orbits to precess around the Earth. At the right combination of altitude and inclination, this precession will be at the same rate of the Earth's orbit around the sun, resulting in the orbit appearing to be fixed relative to the sun. This means that the local time under the spacecraft is constant year round, for a given latitude. In addition, by carefully selecting the altitude of the orbit such that the ratio of orbit period to day length is rational, the ground track of the orbit will be periodic, looping back on itself after a certain number of days. Combined, these two properties result in a spacecraft that will return to each observed location at a periodic interval, and that each observation of a location will always be at the same time of day. This controls two variables in Earth observation missions, and is often a desired behavior. For simplicity, we took the eccentricity e, right ascension of the ascending node Ω, and starting argument of latitude θ to be 0.

In our numerical analysis, all orbits were modeled as under the influence of Earth's gravity including J_2 effects, but higher order perturbations and three body effects including those of the moon and sun were ignored. Similarly, at an altitude of 747 km, air resistance was assumed to be negligible, and we assume a homogeneous spacecraft geometry, making the relative drag particularly small. While these additional higher order perturbations will also affect the orbit, we expect them to be on the same order of magnitude as the 7.55 mm/orbit drift rate in the along track for the J_2 invariant PROs [186]. Both of these effects can be addressed by normal station keeping methods throughout the mission, and are assumed to be of the same order of magnitude for all deputies, regardless of formation design, and hence are not optimized for.

In order to improve convergence and numerical stability, the median separation was used as a more robust statistic than the mean separation during initial convergence of the genetic algorithm.

Note that several undetermined parameters must be tuned by the designer. Most importantly, the number of agents does not directly enter into the objective function. This was done deliberately to reduce the dimensionality of the optimization problem. Instead of setting the number of agents as a variable to optimize over, the best design for each size formation can be computed, and the trade off between scientific merit versus added complexity and mission cost weighed by the design team or program. Effectively, this is simply allowing the human designers to determine the relative weighting between added agents and added resolution, but we recognize that this evaluation will necessarily change as the mission is developed. Instead of prematurely optimizing this parameter before a spacecraft design is finalized, the trade space for each formation size of interest will be explored. In this analysis, formations of 6 and 4 spacecraft were studied in particular. Similarly, the weighting between the setup cost (in terms of the propellant used) and the science produced, α, is also tuneable by the designer. In practice, the set up cost was found to be nominal, and similar between most formations. For this analysis a value of $\alpha = 1$ was used for simplicity, as this was not found to have a significant effect on the results of the optimization. Instead, a large setup cost was usually indicative of a infeasible orbit or a bad initial condition for the optimization algorithm.

E.3 Real-time Orbit Selection for Inspection Implementation Details

Here we provide additional implementation details for our numerical simulations. All simulations were run in Python on a 10-core M1 2021 Macbook Pro with 16 GB of RAM.

PRO Details

Our chief satellite is the target of interest, a 10 meter sphere in the same orbit we considered for the DARTS mission (altitude 747 km, eccentricity 0, inclination 98.4°, right ascension of the ascending node 0°, argument of periapsis 0°). All simulations start at the ascending node.

To generate a PRO library C, we fix the size of the PRO candidate space n. We uniformly distribute n points on a sphere according to a Fibonacci lattice. These points represent the initial conditions of orbits in local vertical, local horizontal (LVLH) coordinates $(x_i(0), y_i(0), z_i(0))$ for $i = 1, \ldots, n$. We take all initial positions to be on a sphere of fixed radius of 1 kilometer from the target satellite. The initial velocities $(\dot{x}_i(0), \dot{y}_i(0), \dot{z}_i(0))$ are found by solving for the J_2-invariant orbits corresponding to $(x_i(0), y_i(0), z_i(0))$ [186]. These initial positions and velocities are forward integrated with J_2-perturbed orbital dynamics over a single orbital period to solve for the configuration $(x_i(t), y_i(t), z_i(t), \dot{x}_i(t), \dot{y}_i(t), \dot{z}_i(t))$ at each time step t throughout the orbit. The configurations $(x_i(t), y_i(t), z_i(t))$ are the sensor poses used to evaluate the information cost $H(\mathcal{P})$. We use an orbital period of 6000 seconds, with a time step of 5 seconds. This corresponds to an orbit altitude of approximately 750 kilometers.

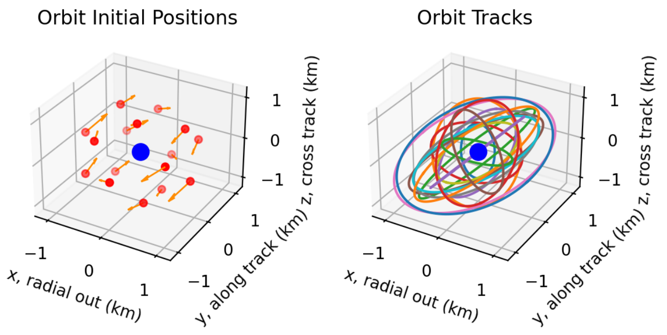


Figure E.2: **The orbits in the size** n = 16 **PRO library.** On the left are the initial positions and velocities of the candidate PROs in LVLH coordinates. On the right are the trajectories of all sixteen PROs after forward integrating for one orbit.

For thee PRO library sizes of $n = 80, 100, 120, 140,$ and 160, we again uniformly distribute n initial conditions on a sphere. We perturb the radius of the initial condition sphere to a 1D Gaussian with mean 1 km and standard deviation 0.2 km. For uniformity across simulations, each simulation ran on a size n library use the same randomly-perturbed space. This perturbation serves to create a less smooth search space, so that we can examine the behavior of the MCTS and greedy policies on a larger and more difficult search problem.

Visibility Checking Details

To find the information cost of viewing a point of interest s on the surface of the target satellite, we evaluate Eq. (7.4) for every configuration p that each deputy satellite takes around the target and perform the reciprocal sum with the baseline variance w. To model visibility of a point of interest s from a deputy satellite position p, we project a 10-degree half-angle cone radially outward from s and check for intersections with p. For simplicity, we assume the sensor is always oriented at the target. A visualization of the visibility-checking is shown in Fig. E.3. We then sum the point-of-interest contribution for each point s on the surface of the target. The resulting sum is the total information cost H of a set of PROs. We refer to this as a cost to indicate that lower values of H represent larger amounts of information, and therefore more favorable configurations for viewing the target. As shown in Eq. (7.4), completely unseen points of interest contribute cost w to the total information cost H.

Optimal and Greedy Orbit Assignment Details

The optimal set of T PROs is found by checking every combination $\binom{n}{T}$ set of PROs, and selecting the set with the lowest information cost. For the libraries of size $n = 8, ..., 20$, the formation size is $T = 4$. For the larger libraries of size $n = 80, 100, 120, 140,$ and 160, the formation size is $T = 10$. Finding the global optimal set of PROs would take intractably long for these larger libraries, as it would require searching over at least 10^{12} sets. For this reason, we considered only the performance of the MCTS policy and the greedy policy for these libraries.

The greedy assignment of PROs is done sequentially. Beginning with an empty set, all PROs in the library are ordered by evaluating the information cost for each individually, and the lowest-cost PRO is chosen. For the next iteration, we select a PRO that when combined with the first PRO results in the lowest-cost two orbit formation. This process is repeated until T PROs are chosen.

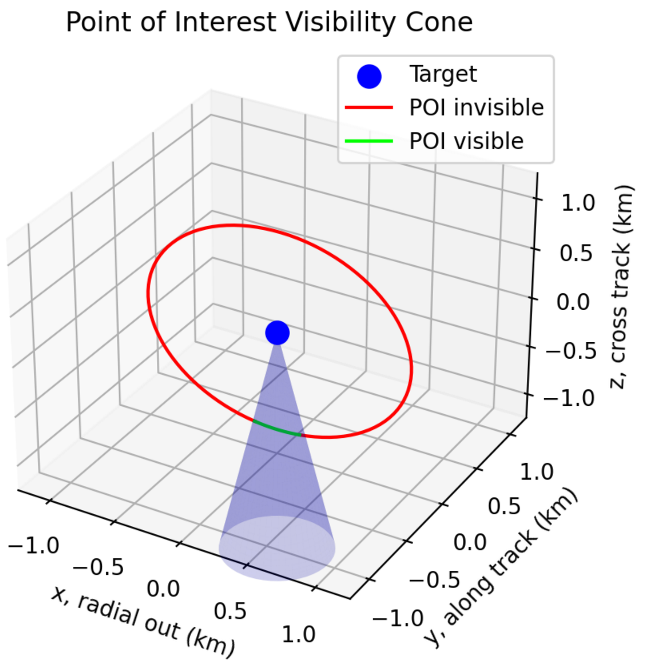


Figure E.3: **Point of Interest visibility cone visualized.** The target is shown as a blue sphere at the origin of the image, with the visibility cone of a point of interest on the negative-z side projected downward. Any deputy satellite in the PRO, shown in red and green, only sees the point of interest for the green portion of its orbit.

Algorithm Details

When working with PRO libraries of size $n = 20$ or less, we use $M = 100$ iterations. For the large libraries, $n = 80$ and larger, we use $M = 1000$ iterations. The algorithm represents possible sets of PROs in a tree structure, where each node of the tree is a set of orbits $\mathcal{P}$ of up to size T. An edge connecting a parent node to a child node represents the addition of one PRO to the parent node's set. As such, the number of PROs in a set corresponds to that node's depth in the tree.

Each node $\mathcal{P}$ also contains five pieces of data used for tree traversal. The first two are values used in standard Monte-Carlo tree search: total cost $\mathcal{P}.Q$ of the terminal nodes below and total number of visits $\mathcal{P}.N$ to the branch starting at $\mathcal{P}$. The next three pieces of information are included as this Monte-Carlo tree search is relatively shallow and wide ($T << n$). The as-of-yet best set of PROs found during the search and the associated value are stored as $\mathcal{P}$.best_set and $\mathcal{P}$.best_cost. Updated during the backpropagation step, these sets allow the algorithm to quickly report the most-optimal set of PROs upon termination. The final component is a boolean flag, denoted $\mathcal{P}$.complete, storing whether every node is fully expanded below it. A node at depth d is fully expanded if it is either terminal ($d = T$) or if it has $n - d$ children (and therefore all PROs remaining in the library have been tried).

We use the standard UCT implementation to recursively select a node to expand [35], with an exploration parameter of $c = 1.2$. In our algorithm's notation, this is given as:

$$\text{UCT}(\mathcal{P}, \mathcal{R}, c) = \mathcal{R}.Q/\mathcal{R}.N - c\sqrt{\frac{\log(\mathcal{P}.N)}{\mathcal{R}.N}} \tag{E.1}$$

Monte-Carlo tree search expands nodes randomly, to evaluate information cost and time-to-run, we simulated orbit assignment using five randomly-generated seeds and average the costs and times over five runs.

Algorithm 9: Monte-Carlo Tree Search

```
Input: $C$: Set of PRO candidates
Input: $T$: Target set size
Input: $c$: Exploration/exploitation parameter
Input: $M$: Number of algorithm iterations
Output: $\mathcal{P}^*_{MCTS}$: The most optimal set of PROs found in $M$ iterations
1  Function MCTS($C$, $T$, $c$, $M$):
2      Initialize empty tree $\mathcal{T}$;
3      for $k = 1 \dots M$ do
           /* Search down tree until reaching un-expanded node */
4          $\mathcal{P} \leftarrow$ SelectUntilUnexpandedNode($\mathcal{T}$.root, $T$, $c$);
           /* Expand node */
5          $\mathcal{P}' \leftarrow$ Expand random un-expanded child of $\mathcal{P}$;
           /* Add new node to tree */
6          $\mathcal{T}$.append($\mathcal{P}'$);
           /* Rollout node to end of decision process */
7          $\mathcal{P}'_{\text{full}} \leftarrow \mathcal{P}'$ filled with random PROs;
           /* Find value of rolled-out node */
8          $V \leftarrow H(\mathcal{P}'_{\text{full}})$;
9          $\mathcal{P}'.Q \leftarrow V$;
10         $\mathcal{P}'.N \leftarrow 1$;
           /* Backpropagate value/visit count */
11         while $\mathcal{P}$ is not root node do
               /* Update cost */
12             $\mathcal{P}.Q$ += $V$;
               /* Update visit counts */
13             $\mathcal{P}.N$ += 1;
14             if $V \leq \mathcal{P}.best\_cost$ then
15                 $\mathcal{P}$.best_set, $\mathcal{P}$.best_cost $\leftarrow \mathcal{P}', V$;
               /* Update completeness flag */
16             if Tree below $\mathcal{P}$ is complete then
17                 $\mathcal{P}$.complete $\leftarrow$ True;
               /* Step up the tree */
18             $\mathcal{P} \leftarrow \mathcal{P}$.parent;
19     return $\mathcal{T}$.root.best_set;
```

Algorithm 10: SelectUntilUnexpandedNode

```
Input: $\mathcal{P}$: Node to begin UCT-search at
Input: $T$: Target set size
Input: $c$: Exploration/exploitation parameter
Output: $\mathcal{P}$: Node selected to perform expansion
1 if $\mathcal{P}$ has un-expanded children then
      /* Select this node                                    */
2     return $\mathcal{P}$;
  /* Choose best child by UCT (Eq. eqrefeq:UCT) and recurse */
3 else
4     best_child $\leftarrow \underset{\mathcal{R} \in \mathcal{P}\text{.incomplete\_children}}{\arg\max} \left\{ \text{UCT}(\mathcal{P}, \mathcal{R}, c) \right\}$;
5     return $\mathit{SelectUntilUnexpandedNode}(\mathit{best\_child}, T, c)$;
```