Do Robots Dream of Random Trees? Monte Carlo Tree Search for Dynamical, Partially Observable, and Multi-Agent Systems

Thesis by
Benjamin Rivière

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy In Aeronautics

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2024
Defended May 29, 2024

Benjamin Rivière
ORCID: 0000-0002-0597-5400

ACKNOWLEDGEMENTS

To my advisor: Soon-Jo Chung. Thank you for always supporting me and challenging me to be technically excellent and intellectually unique. I have learned much about research and life from you, and I am very grateful to have been your PhD student.

To my committee: Yisong Yue, Fred Hadaegh and Sergio Pellegrino. Thank you for your invaluable guidance throughout my studies.

To my sponsors: Aerospace Corporation, Supernal, DARPA, Jet Propulsion Laboratory, and United Technologies. Thank you for your feedback and financial support.

To my principal collaborators: Wolfgang Hoenig, James Ragan and John Lathrop. I am very lucky and thankful to have worked with and learned from all of you.

To the ARCL lab: Matt Anderson, Sorina Lupu, Rebecca Foust, Xichen Shi, Yashwanth Nakka, Karena Cai, Kai Matsuka, Guanya Shi, Hiroyasu Tsukamoto, Michael O'Connell, SooJean Han, Ellande Tang, Connor Lee, Nikhil Ranganathan, Fengze Xie, Lu Gan, James Preiss, Vincenzo Capuano, Patrick Spieler, Salar Rahili, Kyunam Kim, Anthony Fragoso, Jedidiah Alindogan, Hannah Grauer, Xingxing Zuo, Anthony Clark, and Josh Cho. Thank you all for making the lab a supportive and stimulating environment, I think of you all as my second "work" family.

To my partner Tara: Thank you for always supporting me and for sharing your beautiful sense of humor.

To my brother and my parents: Thank you for always supporting me and believing in me.

ABSTRACT

Autonomous robots are poised to transform various aspects of society, spanning transportation, labor, and scientific space exploration. A critical component to enable their capabilities is the algorithm that interprets sensor data to generate intelligent planned behavior. Although reinforcement learning methods that train parameterized policies offline from data have shown recent success, they are inherently limited when robots inevitably encounter situations outside their training domain. In contrast, optimal control techniques, which compute trajectories in real-time using numerical optimization, typically yield only locally optimal solutions.

This research endeavors to bridge the gap by developing algorithms that compute trajectories in real-time while converging towards globally optimal solutions. Building upon the Monte Carlo Tree Search (MCTS) framework—a stochastic tree search method that simulates future trajectories while balancing exploration and exploitation—the research focus is twofold: (i) constructing an efficient discrete representation of continuous systems in a decision trees, and (ii) searching on the resulting tree while balancing exploration and exploitation to achieve global optimality.

The study spans theoretical analysis, algorithmic design, and hardware demonstrations across dynamical, partially observable, and multi-agent systems. By addressing these critical questions, this research aims to advance the field of autonomous robotics, enabling the deployment of intelligent robots in complex and diverse environments.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] James Ragan, Benjamin Rivière, and Soon-Jo Chung. "Dreaming to Disambiguate: Safe Fault Estimation via Active Sensing Tree Search". In: *(Review at Science Robotics)* (2024).
B.R. co-led project conceptualization and supported algorithm design, theoretical analysis, simulation development, hardware experiments, and paper writing.

[2] Benjamin Rivière*, John Lathrop*, and Soon-Jo Chung. "Monte Carlo Tree Seach for Dynamical Systems with Spectral Expansion". In: *(Review at Science Robotics)* (2024).
B.R. co-led project conceptualization, algorithm design, theoretical analysis, simulation, and paper writing.

[3] James Ragan*, Benjamin Rivière*, and Soon-Jo Chung. "Bayesian Active Sensing for Fault Estimation with Belief Space Tree Search". In: *AIAA SciTech* (2023). DOI: 10.2514/6.2023-0874.
B.R. co-led project conceptualization, algorithm design, theoretical analysis, simulation, and paper writing. **Best Graduate Student Paper Award in Guidance, Navigation and Control at AIAA 2023.**

[4] Benjamin Rivière and Soon-Jo Chung. "H-TD2: Hybrid Temporal Difference Learning for Adaptive Urban Taxi Dispatch". In: *IEEE Transactions on Intelligent Transportation Systems* (2021), pp. 1–10. DOI: 10.1109/TITS.2021.3097297.
B.R. led project conceptualization, algorithm design, theoretical analysis, simulation, and paper writing.

[5] Benjamin Rivière, Wolfgang Hönig, Matthew Anderson, and Soon-Jo Chung. "Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environments". In: *IEEE Robotics Automation Letters* 6.4 (2021), pp. 6868–6875. DOI: 10.1109/LRA.2021.3096758.
B.R. led project conceptualization, algorithm design, theoretical analysis, simulation, hardware experiments, and paper writing.

[6] Benjamin Rivière, Wolfgang Hönig, Yisong Yue, and Soon-Jo Chung. "GLAS: Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning With End-to-End Learning". In: *IEEE Robotics Automation Letters* 5.3 (2020), pp. 4249–4256. DOI: 10.1109/LRA.2020.2994035.
B.R. led project conceptualization, algorithm design, theoretical analysis, simulation and hardware experiment, and paper writing. **Honorable Mention for Best Paper Award at IEEE RA-L 2020.**

The * denotes equal contribution.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Chapter 1

INTRODUCTION

In this chapter, we present the sequential decision-making problem and its solutions formalized with dynamic programming and calculus of variations. Then, we introduce Monte Carlo Tree Search and show it is an approximate multi-step Bellman operator and a member of the generalized policy iteration (GPI) reinforcement learning family. Having established the context of reinforcement learning and optimal control, we outline the remainder of the thesis.

1.1 Decision-Making Problem

The components of autonomous decision making are shared by reinforcement learning and optimal control: the world is described by a state x, that evolves in time under some transition F depending on the action taken by the robot u and the robot's goal is to find the policy π that maximizes its expected accumulated reward r over time, known as the value V^π.

This can be formulated as follows:

$$x_k = F(x_{k-1}, u_k) \tag{1.1}$$

$$r_k = R(x_k, u_k) \tag{1.2}$$

$$V^\pi(x, t) = \sum_{k=t}^{K} \gamma^{k-t} r_t + \gamma^{K-t} D(x_K) \tag{1.3}$$

where the policy selects the action at each timestep, $u_k = \pi(x_{k-1})$ and k is the timestep, $\gamma \in [0, 1)$ is the discount factor, K is the horizon and D is the terminal value boundary condition.

There are many variations to the sequential decision-making problem: continuous time, partial state information, multi-agent settings, stochastic dynamics, stochastic policy, infinite horizon, and others. The notation, which is consistent throughout the thesis, is introduced now to describe the reinforcement learning and optimal control perspectives. The components are visualized in Fig. 1.1.

1.2 Principle of Optimality-Based Solutions

Richard Bellman coined the principle of optimality in the following statement:

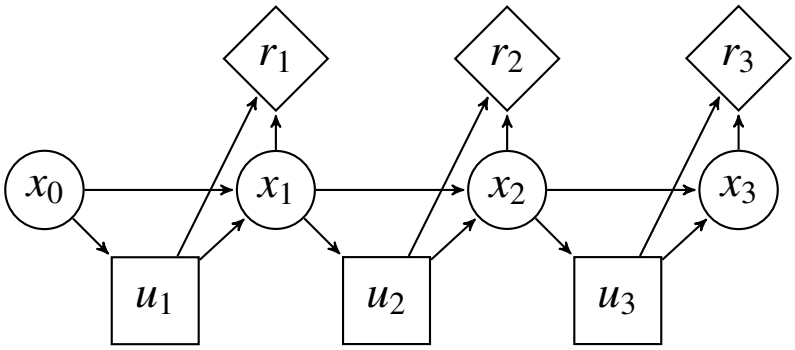


Figure 1.1: The components of a sequential decision-making problem.

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

The principle of optimality applied to the optimal value function is called dynamic programming [1]. In continuous time and space, dynamic programming can be used to derive the Hamilton-Jacobi-Bellman partial differential equation:

$$\frac{\partial V^*}{\partial t} + \max_u \left\{ \frac{\partial V^*}{\partial x} F(x,u) + R(x,u) \right\} = 0 \tag{1.4}$$

subject to the boundary condition $V^*(x,K) = D(x)$.

In discrete time and space and in the infinite horizon, dynamic programming can be used to derive the Bellman equations [2]:

$$V^*(x) = \max_u \{R(F(x,u)) + \gamma V^*(F(x,u))\} \tag{1.5}$$

The Bellman equations are the fixed point of the Bellman operator $\mathcal{T}$, which is a contraction mapping for discounted problems:

$$(\mathcal{T}V)(x) \equiv \max_u \{R(x,u) + \gamma V(F(x,u))\} \tag{1.6}$$

$$|\mathcal{T}V_1 - \mathcal{T}V_2| \leq \gamma |V_1 - V_2| \tag{1.7}$$

The contraction of the Bellman operator is the foundation of Generalized Policy Iteration (GPI), an important concept in dynamic programming that unifies many reinforcement learning techniques. In GPI, the algorithm keeps track of a policy and value function and iteratively updates them with two processes: policy evaluation (estimate the value function of the current policy) and policy improvement (make the policy greedy with respect to the current value function).

Overall, dynamic programming provides a theoretical foundation for global solutions and provides general intuition for convergence of dynamic programming-based reinforcement learning. The resulting numerical algorithms have polynomial complexity with respect to the number of states and actions, and are effective when the state and action spaces are discrete and finite. However, robots exist in a high-dimensional continuous world, and therefore naive application of dynamic programming (e.g., discretize then use dynamic programming) are not practical because of the curse of dimensionality. For example, the number of discrete elements of a uniformly discretized n-cube is η^n where η is the number of points per dimension and n is the dimension.

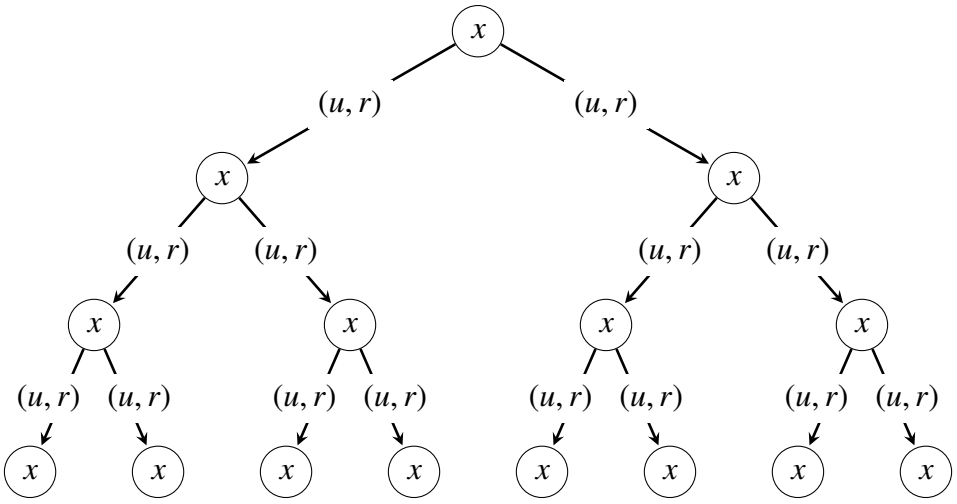
The principle of optimality can also be applied to the optimal trajectory and analyzed with calculus of variations [3]. The basic calculus of variations problem is to find the curve that, given boundary conditions, minimizes the its path-integrated Lagrangian. The Euler-Lagrange equations are derived from the first-order optimality condition of path minimization and, using Hamilton's principle of least action, these equations can be used to describe the motion of mechanical systems. Calculus of variations also provides a general solution to very old problems like Dido's isoperimetric problem (850 BC) and the Brachistochrone (1696 AD). In optimal control, the reward function is the Lagrangian, and computing first-order necessary conditions of the optimal value function leads to a result known as Pontryagin's maximum principle.

Overall, calculus of variations provides the theoretical foundation for necessary conditions of solutions in continuous space. However, numerical solutions that use the calculus of variations results typically only converge to locally-optimal trajectories and can be arbitrarily suboptimal.

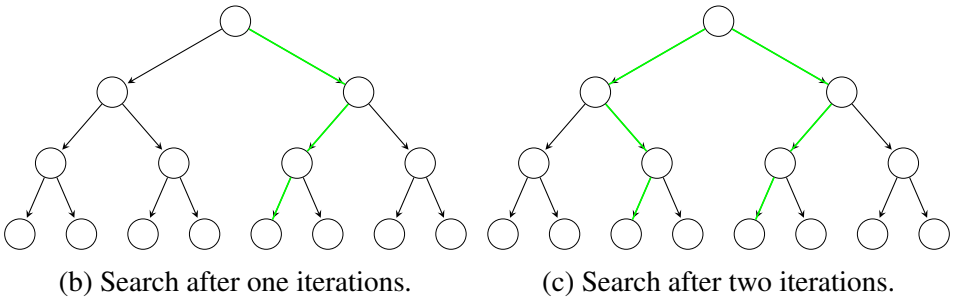
The principle of optimality is not the only way to define solutions to the sequential decision-making problem. For example, policy gradient [4] is an empirically successful and popular method, and its theoretical foundation is a result that shows trajectory data can be used to estimate the gradient of the optimal value function in an unbiased manner. This policy gradient method is classified as stochastic optimization, and is not the focus of this thesis, mentioned here only for context.

1.3 Monte Carlo Tree Search

The decision-making problem can also be represented as a tree search as originally discussed in [5], and shown in Fig. 1.2. Monte Carlo Tree Search (MCTS) is a

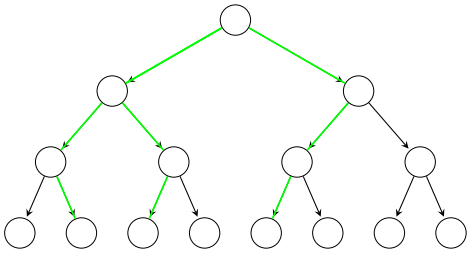


(a) Decision-making problem represented as a tree.

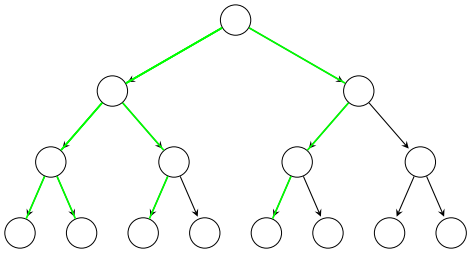


(b) Search after one iterations.

(c) Search after two iterations.



(d) Search after three iterations.



(e) Search after four iterations.

Figure 1.2: Decision-making problem is represented as a tree, and then MCTS is applied to the resulting tree.

family of methods that search on the resulting tree by sampling future trajectories while balancing exploration of unvisited space and exploitation of already-visited highly valued space [6]. The most common MCTS variant is the Upper Confidence Bound for Trees (UCT) [7], which was famously used in combination with deep learning in the AlphaZero algorithm [8] to achieve superhuman performance in the game of Go.

The convergence result of UCT is that the root node's value estimate converges to the optimal value with the number of trajectories in the tree. When UCT rolls out

a random trajectory, it gathers samples of the value conditioned on the tree policy and, if the tree policy was static, the mean of these samples would converge to the true expectation exponentially fast from Hoeffding's concentration inequality. However, because the goal is to converge to the optimal policy, the tree policy biases towards areas of high reward, creating a non-stationary data distribution and adding recursive complexity to the convergence analysis. The authors claim this non-stationarity can be completely addressed by increasing the exploration constant of the classic online-learning Upper Confidence Bound result [9]. Although the UCT algorithm is empirically very successful, its theoretical result is disputed in literature [10, 11] and analyzing Monte Carlo Tree Search algorithms is an active area of research.

Another important analysis of MCTS is a GPI-based interpretation of AlphaZero's self-play and training iteration [12]. In particular, MCTS acts as a approximate multi-step Bellman Operator: first, we introduce an one-step Bellman operator where the action is selected with respect to an approximate value function, $\tilde{V}$. This can be extended to a multi-step Bellman operator where the explicit maximization occurs over a sequence of actions, then greedy with respect to the approximate value function.

$$(\tilde{\mathcal{T}}V)(x) \equiv \max_u R(x,u) + \gamma \tilde{V}(F(x,u)) \tag{1.8}$$

$$(\tilde{\mathcal{T}}_l V)(x_0) \equiv \max_{u_1,\ldots,u_l} \left\{ \sum_{k=0}^{l-1} \gamma^k R(x_k,u_k) + \gamma^l \tilde{V}(x_l) \right\} \tag{1.9}$$

In the AlphaZero setting, the approximate value function $\tilde{V}$ is the learned neural network and the l-step is the depth of the tree. The significance of l-step lookahead maximization is that by increasing the value of l, we may require a less accurate value approximation to obtain good performance, at the cost of a more computationally expensive look-ahead phase. This interpretation is used to explain the improved learning stability of AlphaZero in challenging planning problems.

MCTS is significant for two reasons: First, it is only algorithm explicitly designed for real-time generation of globally optimal solutions in large search spaces. Second, it can be combined with deep learning in the principled theoretical framework to create a two-process model for decision making that smoothly transitions between cheap data-driven memorization with neural networks and expensive causal reasoning with tree search. However, MCTS is only well defined for finite action

spaces and, for the same reason as other dynamic programming techniques, its performance suffers when applied directly to continuous robot optimal control.

1.4 Thesis Outline

In the context of reinforcement learning and optimal control, the goal of my research is to develop algorithms with real-time computation of globally optimal solutions. With this in mind, my research focuses on two questions: (i) constructing an efficient discrete representation of continuous systems in a decision trees, and (ii) searching on the resulting tree while balancing exploration and exploitation to achieve global optimality. Chapters 2 and 3 are focus on the first question in the settings of dynamical and partially observable systems. Chapter 4 develops the second question and uses decentralized data-driven heuristics to control the tree growth for N-player games. Chapter 4 builds on the learning techniques developed in Chapter 5.

In Chapter 2, we develop an MCTS algorithm for continuous dynamical systems. Our method, called Spectral Expansion Tree Search (SETS), is MCTS with a novel nodal expansion that generates trajectory segments using the spectrum of the locally-linearized system's controllability Gramian. SETS is equivalent to running UCT on an alternative problem, where this alternative problem has a low-complexity discrete action space and its optimal value function is of bounded distance to that of the original problem. Using this insight, we are the first to prove UCT's global optimality convergence in continuous space. We validate the algorithm with three real-time hardware demonstrations not directly solvable with other methods: (i) a quadrotor plans through wind field using deep neural network dynamic models, (ii) a tracked vehicle assists a driver through an adversarial concourse, and (iii) a team of tethered spacecraft catch and redirect an uncooperative target. We further validate on a simulated aerodynamic glider environment, where the empirically observed value convergence rates match the rates predicted by theory. We also show that SETS vastly outperforms comparable state-of-the-art techniques.

In Chapter 3, we develop an MCTS algorithm for planning with uncertainty, specialized to the active diagnosis problem for stochastic systems subject to sensor and actuator failure. Our method, called Safe Fault Estimation with Active Sensing Tree Search (s-FEAST), uses a marginalized filter to compute an otherwise intractable Bayesian update at each nodal expansion. Similarly to the previous chapter, our analysis shows sFEAST is equivalent to running UCT on an equiva-

lent belief-space decision-making problem. Although originally derived for fully observeable systems, the UCT convergence proof can also be applied to the belief-space problem, and we use this fact to prove the convergence of our algorithm to the globally optimal solution. We extend our method to satisfy chance constraints for general probability distributions with concentration inequalities. Using our robotic spacecraft simulator, we experimentally validate s-FEAST by safely and successfully performing fault estimation while on a collision course with a model comet. These results are further validated through extensive numerical simulations demonstrating s-FEAST's superior performance.

In Chapter 4, we develop an MCTS algorithm for N-player team games for robots. Our search algorithm, called Neural Tree Expansion (NTE), uses decentralized data-driven heuristics to guide the tree search. We use a second algorithm to train the heuristics offline and, at a high level, the training is similar to AlphaZero and GPI dynamic programming: initializes the heuristics, creates a dataset of NTE self-play samples, trains the heuristics, and iterate. However, the multi-robot problem is different from traditional board game problems and therefore our algorithm diverges from AlphaZero to satisfy critical differences: decentralized evaluation, real-time evaluation, and continuous action space. We validate NTE on multiple simulated environments, including an experiment that shows our solutions converge to the analytical solution of the homicidal chauffer differential game. We also validate NTE on hardware for the three vs three quadrotor Reach-Target-Avoid game.

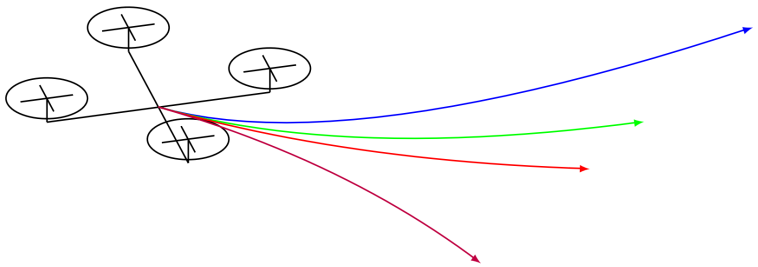
The previous chapter relies on machine-learning techniques developed in Chapter 5. There, we develop Global-to-Local safe Autonomy Synthesis (GLAS), an imitation learning technique for training decentralized policies from centralized data while providing end-to-end safety guarantees. Our approach combines the advantage of centralized planning of avoiding local minima with the advantage of decentralized controllers of scalability and distributed computation. In particular, our synthesized policies only require relative state information of nearby neighbors and obstacles, and compute a provably safe action. Our approach has three major components: i) we generate demonstration trajectories using a global planner and extract local observations from them, ii) we use deep imitation learning to learn a decentralized policy that can run efficiently online, and iii) we introduce a novel differentiable safety module to ensure collision-free operation, thereby allowing for end-to-end policy training. Our numerical experiments demonstrate that our policies have a 20 % higher success rate than optimal reciprocal collision avoidance, ORCA, across a

wide range of robot and obstacle densities. We demonstrate our method on an aerial swarm, executing the policy on low-end microcontrollers in real-time.

Finally, in Appendix C, we present a multi-agent reinforcement learning algorithm called Hybrid Temporal Difference Learning for Taxi Dispatch (H-TD^2). Although this algorithm and analysis shares themes of manipulating problem representation for efficient dynamic programming, it does not use a tree search algorithm and therefore we choose to exclude it from the main body of the thesis.

Chapter 2

MONTE CARLO TREE SEARCH (MCTS) FOR DYNAMICAL SYSTEMS



How to construct discrete trees from continuous systems?

This chapter is based on the publication:

> Benjamin Rivière*, John Lathrop*, and Soon-Jo Chung. "Monte Carlo Tree Seach for Dynamical Systems with Spectral Expansion". In: *(Review at Science Robotics)* (2024).
>
> The * denotes equal contribution.

2.1 Motivation

Endowing robots with high-performing and reliable autonomous decision making is the ultimate goal of robotics research and will enable applications such as sea, air, and space autonomous exploration, self-driving cars, and urban air mobility. A conventional metric of robotic autonomy is formalized through the decision making problem: how well do the robot's actions optimize an objective over time, subject to dynamics and environmental effects?

This vision of robotic autonomy remains elusive because exactly solving the continuous-space decision making problem in high-dimensional systems has a large computational complexity, as originally shown by a founder of the field, Richard Bellman [2]. In light of this complexity, many autonomous robots in deployment avoid directly solving a general decision making problem and instead exploit particular problem structure for computational benefits. For example, motion planning can be solved with sampling-based methods [3–5], trajectory optimization can be solved

with convex optimization [6, 7], and high-level discrete decision making can be solved with value iteration [8]. These methods can also be combined hierarchically for complex behavior, such as autonomous multi-agent inspection of spacecraft [9], robotic manipulation [10, 11], and self-driving urban vehicles [12, 13]. Problem-specific solutions are well-understood, can be made computationally efficient, and have proven to be effective. However, these solutions are limited because they cannot be easily transferred to new problems, adding an expanding burden to the designer for multi-task autonomy. Additionally, there exist problems that cannot easily be decomposed into computationally tractable sub-problems.

A competing approach uses reinforcement learning to train optimal policies from trajectory data. Unlike the aforementioned methods, this approach is a generalized procedure that can be applied directly to a broader class of problems. This property enables important new capabilities such as drone racing [14], helicopter flight [15], grasping [16] and bipedal locomotion [17]. However, these methods typically require an offline training phase that limits deployment in new or changing environments. In addition, these black-box approaches are unexplainable and provide limited guarantees on optimality, stability, or robustness.

Alternative reinforcement learning approaches use tree data structures [18–20] that strategically explore simulated future trajectories from the current state. These methods are collectively known as Monte Carlo Tree Search (MCTS) [21]. In contrast with learning methods with offline training, MCTS generates near-optimal solutions in real-time: given a cost or reward function, the goal is to return the best possible plan with the computational budget available. Whereas the tree's nodes and edges are naturally defined for discrete spaces, the continuous space of robotics presents new challenges. Uniform spatial and temporal discretization of continuous spaces leads to very large trees and slow convergence rates: a poor discrete *representation* of the underlying continuous problem.

In this work, we present Spectral Expansion Tree Search (SETS), a new algorithm that finds globally optimal solutions to the deterministic continuous-space decision making problem in real-time. The new capability of SETS is enabled through efficient representation of the continuous space of future possibilities by constructing the tree's edges with trajectories that drive the system along its natural motions, a concept formalized through the spectrum of the locally linearized controllability Gramian. Without further assumptions on the dynamics or reward, the transformed low-complexity form is solved with MCTS. The SETS tree is visualized

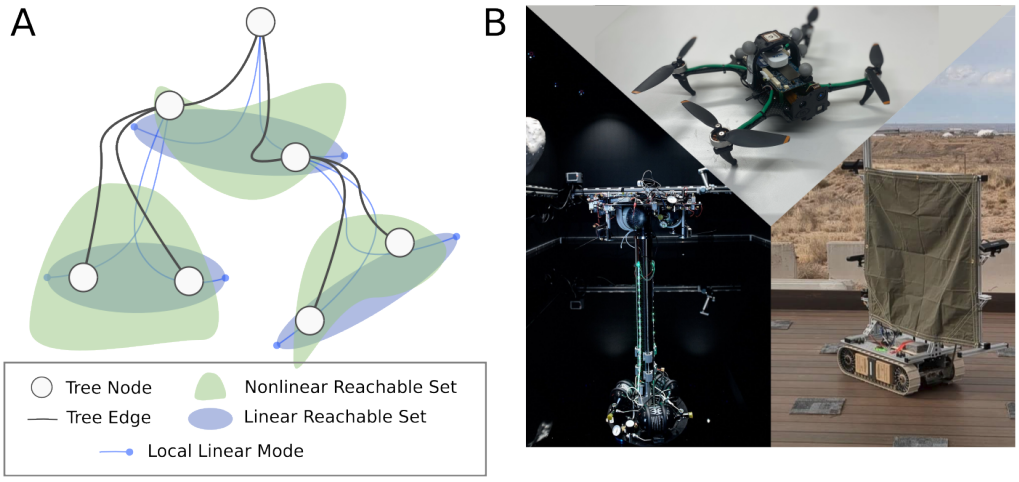


Figure 2.1: SETS algorithm concept overview and wide-applicability in robotics, spanning ground, aerial, and space domains.

in Fig 2.1A. Compared to a direct discretization, our method reduces the branching factor and number of decisions in the time horizon, leading to an exponential reduction in tree size. Our analysis proves fast convergence to a bound of the globally optimal solution for decision making problems with deterministic and differentiable dynamics, state-dependent and Lipschitz rewards, and continuous state-action spaces.

Our hardware and simulation experiments showcase a diverse set of "discovered-not-designed" behaviors generated in real-time for various robot dynamics and objectives: (i) a quadrotor solves a dynamically constrained Traveling Salesman Problem to quickly monitor multiple targets in a windy arena, selectively traversing wind gusts and avoiding floating ball obstacles; (ii) a tracked vehicle shares control with a driver through a concourse of ramps, chicane, and sawtooth tracks subject to adversarial degradation; (iii) a team of spacecraft use a net to capture and redirect an uncooperative target in a frictionless environment; and (iv) in simulation, a glider experiencing aerodynamic drag detours into a thermal to extract energy from the environment and survive long enough to achieve its directed task. We use this last experiment as a case study to empirically validate the theoretical result, compare our method to state-of-the-art baselines, and tune parameters in a systematic and informed procedure.

2.2 Problem Formulation

We consider a Markov Decision Process (MDP) [22], an abstract problem description written as a tuple of components: $\langle X, U, F, R, D, \Omega, K, \gamma \rangle$. Here $X \subseteq \mathbb{R}^n$ is a compact state space, $U \subseteq \mathbb{R}^m$ is a compact action space, and $F : X \times U \to X$ are the discrete-time dynamics. $R : X \times U \to [0, 1]$ is the stage reward and $D : X \to \mathbb{R}_{\geq 0}$ is the terminal reward. $\Omega \subseteq X$ is a set of unsafe states, $K \in \mathbb{N}$ is the time horizon, and $\gamma \in [0, 1)$ is the discount factor.

At an initial state x_0, the decision making problem is to select a sequence of actions that maximizes the sum of the stage reward plus the terminal reward, subject to the dynamics and state/action constraints:

$$V^*, x^*_{[K]}, u^*_{[K]} = \underset{x_{[K]},\, u_{[K]}}{\operatorname{argmax}} \sum_{k=0}^{K-1} \gamma^k R(x_{k+1}, u_{k+1}) + \gamma^K D(x_K) \tag{2.1}$$

$$\text{s.t.} \quad x_k = F(x_{k-1}, u_k), \quad x_k \in X \setminus \Omega, \quad u_k \in U, \quad \forall k \in [1, K]$$

A bracket subscript indicates a sequence, e.g. $x_{[K]} = [x_1^\top, x_2^\top, \ldots, x_K^\top]^\top \in \mathbb{R}^{nK}$.

This framework of decision making over a horizon is the same setting considered by the reinforcement learning community and in optimal control. The space of problems we consider is focused on smooth, deterministic, and nonlinear dynamics with continuous state and action spaces.

2.3 Spectral Expansion Tree Search

Monte Carlo Tree Search

The pseudocode for SETS is shown in Algorithm 1. SETS performs a Monte Carlo Tree Search (MCTS) with a specialized nodal expansion operator, defined as Spectral Expansion in the pseudocode. We briefly summarize the procedure of MCTS, with more in-depth descriptions available in the literature [21]: while the robot has remaining computational budget, simulate future state trajectories from the current world state forward to the horizon of the MDP. At each node, the tree policy selects the best child by balancing exploration of visit counts and exploitation of observed reward. If a node is not fully expanded, generate a new child by taking an action and stepping forward in time. When a simulated trajectory terminates, the accumulated reward and visit count information is backpropagated up the tree to update statistics, and the process iterates.

Whereas the standard MCTS variant [19] uses logarithmic exploration term, SETS uses a polynomial exploration term (see Line 6 of Algorithm 1), because increasing

Algorithm 1: Spectral Expansion Tree Search (SETS)

def SpectralExpansionTreeSearch($x_0, \mathcal{M}, H$)**:**
  /* set root */
  $i_0 = \text{Node}(x_0)$;
  for $\ell = 1, \ldots,$ **do**
    /* initialize path at root then rollout */
    $p = [i_0]$;
    for $d = 1, \ldots, \lceil K/H \rceil$ **do**
      /* best child with random tiebreak */
      $i^* = \arg\max_{i \in C(p[-1])} \frac{V(i)}{n(i)} + c\sqrt{\frac{n(p[-1])}{n(i)}}$;
      if i^* is not expanded **then**
        SpectralExpansion($i^*,\ x_0,\ \bar{u},\ \mathcal{M},\ H$) ;
      if $i^* \in \Omega$ **then** break;
      p.append(i^*) ;
    /* backup */
    for $d = 1, \ldots, |p|$ **do**
      $V(p[d])$ += $\sum_{t=d}^{\lceil K/H \rceil} \gamma^{tH} r(p[t])$;
      $n(p[d])$ += 1 ;
    /* update optimal value estimate */
    $\hat{V}(x_0, \ell) = \max(\hat{V}(x_0, \ell - 1), \sum_{d=1}^{\lceil K/H \rceil} \gamma^{dH} r(p[d]))$;
  yield $\hat{V}(x_0, \ell)$;
def SpectralExpansion($i,\ x_0,\ \bar{u},\ \mathcal{M},\ H$)**:**
  /* compute local linearization */
  $A_k, B_k, c_k = \text{Linearization}(F,\ x_0,\ \bar{u}_{[H]}) \quad \forall k \in [0, H-1]$;
  $\mathbf{z}_{k+1} = L_k(\mathbf{z}_k, u_{k+1}) = A_k \mathbf{z}_k + B_k u_{k+1} + c_k \quad \forall k \in [0, H-1]$;
  /* compute spectrum of normalized controllability Gramian */
  $S = \text{diag}(\{\frac{2}{\bar{u}_j - \underline{u}_j} \mid \forall j \in [1, m]\})$;
  $C = \left[\left(\prod_{k=1}^{H-1} A_k\right) B_0 S,\quad \left(\prod_{k=2}^{H-1} A_k\right) B_1 S,\quad \ldots,\quad A_{H-1} B_{H-2} S,\quad B_{H-1} S\right]$;
  $[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n], [\lambda_1, \lambda_2, \ldots, \lambda_n] = \text{eig}(CC^\top)$;
  /* compute linear reference trajectory to ith mode */
  $\mathbf{z}_H = (-1)^{i\%2} \sqrt{\lambda}_{i//2} \mathbf{v}_{i//2} + \left(\left(\prod_{k=0}^{H-1} A_k\right)\mathbf{z}_0 + \sum_{k=0}^{H-1}\left(\prod_{j=k}^{H-1} A_j\right) c_k\right)$;
  $u_{[H]}^{\text{ref}} = \text{clip}(C^\dagger \mathbf{z}_H, U)$;
  $\mathbf{z}_{[H]}^{\text{ref}} = L^H(\mathbf{z}_0, u_{[H]}^{\text{ref}})$;
  /* track reference trajectory with nonlinear system */
  $M_k = \text{DARE}(A_k, B_k, \Gamma_x, \Gamma_u)$;
  $\mathcal{K}_k = (\Gamma_u + B_k^\top M_k B_k)^{-1} B_k^\top M_k A_k$;
  $u_{[H]} = \{\text{clip}(u_k^{\text{ref}} - \mathcal{K}_{k-1}(x_{k-1} - \mathbf{z}_{k-1}^{\text{ref}}), U) \mid \forall k \in [1, H]\}$;
  $x_{[H]} = F^H(x_0, u_{[H]})$;
  return $(x_{[H]}, u_{[H]}, R^H(x_0, u_{[H]}))$;

the amount of exploration empirically improved the performance of the algorithm. This technique is supported by other MCTS variants, both in theory [23, 24] and practice [25]. Although the dynamics, reward, and spectral expansion operator are deterministic, SETS is a stochastic algorithm because, when multiple children of a node have not been visited, the algorithm randomly selects one. This stochasticity, which is the same as the original MCTS algorithm, is noted in Line 5 of Algorithm 1. We adopt for following notation for the pseudocode: p is the "path", the list of nodes in one rollout, i is a single node, $r(i)$ is the reward to the node, $C(i)$ are its children, and $n(i)$ is its number of visits.

Spectral Expansion Operator

The Spectral Expansion operator, specified in Algorithm 1 Lines 16–29, computes a trajectory of length H. The first step is to compute the linearization and eigendecomposition of the local controllability Gramian. We consider both time-invariant and time-varying linearizations, where the linearized system data is either computed once at the current state and a single nominal control input, or over a time-varying trajectory initialized at the current state, subject to a nominal control sequence. In practice, we found that time-varying linearization produces more stable trajectories, especially for highly agile platforms such as a quadrotor, and we select the unforced dynamics as the default nominal control input, ($\bar{u} \equiv \mathbf{0}$).

From the linearized system, we construct the controllability matrix C, which is a linear mapping from sequences of control actions to the resulting terminal state. This matrix and its associated Gramian are well known in linear control theory [26], with two important properties: *Linear systems have elliptical reachable sets*: the set of states reachable with one unit of control energy are an ellipse parameterized by the spectrum of the Gramian. *Pseudoinverse maps to action sequences:* the pseudoinverse of the controllability matrix applied to a feasible desired terminal state computes the minimum energy control input that drives the system to that state.

In Lines 19-20 of Algorithm 1, the inputs are scaled by their control limits before computing the Gramian. This procedure scales the notion of bounded-energy of the reachable set to the control limits rather than inputs that potentially differ by orders of magnitude. An alternate preconditioning matrix S could be selected here to create a different trade-off in control energy. In addition to informing the input rescaling, the *elliptical reachable sets* property reveals a simple exploration strategy

for linear systems: the vertices of the ellipse, which are computed via the spectrum of the Gramian, form a bounded covering of the reachable set.

In Lines 22-24, we compute the linearized reference trajectory. First, we select the desired state for this branch using the spectrum of the controllability Gramian. We iterate through the modes using the integer floor division and modulus operators, // and %, respectively. Visiting each mode in the plus and minus direction imply a total branching factor of $2n$, where n is the state dimension. The pseudoinverse of the controllability matrix mapped onto the desired state yields a trajectory for the linearized system that, by the *psuedoinverse mapping* property, is minimum energy and reaches the target. However, the notion of minimum and bounded energy is over the entire trajectory and, to verify each individual control actions are valid, we impose the bounded input constraint with a clip operation: given a vector and an interval, the values outside the interval are clipped to the interval edges.

Whereas the reference trajectory is feasible for the locally linearized system, the actual branch trajectory must satisfy nonlinear dynamics. In Lines 25-28, we compute a feedback controller from the Discrete Algebraic Riccati Equation (DARE) [27] and rollout a trajectory of the nonlinear system that tracks the linear reference trajectory to the desired mode. We include a proof of the controller stability in Lemma 5 in the Supplementary Materials.

Heuristics

At the cost of an increased branching factor, the user can add manually designed nodal expansion heuristics to guide the search. In the quadrotor experiments, we use a heuristic to guide the system to the nearest target by including the projection of the nearest target onto the linear set as a branching option. This is similar to goal biased sampling used in the sampling-based motion planning community [28]. For the remaining experiments, we did not use heuristics and relied only on the natural exploration of SETS. In a manner similar to AlphaZero [25] and Neural Tree Expansion [29], it is also possible to incorporate *DNN-based heuristics* to predict the value of the next child state.

The user can also manually decrease the branching factor by prioritizing certain degrees of freedom. In the quadrotor experiment, we found that only searching among the velocity and angular velocity modes led to higher performance. Physically, the system maintains its ability to translate and make attitude adjustments, as the eigenvectors that maximally excite the velocity and angular velocity coordi-

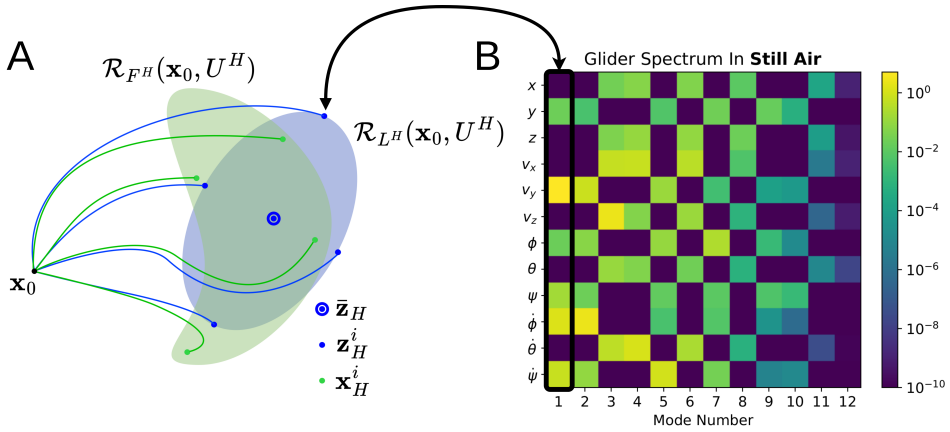


Figure 2.2: The Spectral Expansion Operator and its connection to spectrum of controllability Gramian.

nates also create changes in the position and attitude, respectively. We expect this reasoning to be applicable in many second-order systems, and we apply it in all of our experiments.

2.4 Theoretical Analysis

We outline the proof of our main theoretical result Theorem 1. The full proofs for the following results are in the Supplemental Materials.

We begin with a few assumptions and definitions to enable analysis, namely we consider discounted, smooth, and differentiable-dynamics MDPs. While the analysis relies on these assumptions, the implementation of SETS is more general and only requires access to the MDP. We recall the definition of Lipschitz functions.

Definition 1 *A function* $g : \mathbb{R}^n \to \mathbb{R}^r$ *is Lipschitz continuous if* $\exists L_g \geq 0$ *s.t.* $\forall u, v \in \mathbb{R}^n$, $\|g(u) - g(v)\| \leq L_g \|u - v\|$.

The first assumption is about the smoothness of the dynamics: we assume the dynamics function F is class C^2 - it is twice differentiable, and those derivatives are continuous. Notably, this implies the dynamics are Lipschitz and have Lipschitz gradients, with Lipschitz constant L_F and $L_{\nabla F}$, respectively. Although our analysis requires twice differentiable functions, our implementation only requires first derivatives, which can be computed from the dynamics function using finite differences.

The second assumption restricts the reward structure of the considered class of MDPs: we assume the reward function is only a function of state. Furthermore,

we assume the reward function is Lipschitz continuous with constant L_R. The requirement that the reward be a function of state is not burdensome. An integral control scheme can be set up to include the inputs as part of the state. Additionally, our analysis can be extended for reward functions that include inputs at the cost of extra complication of treating Lemma 2.

The final assumption is that the input set is bounded inside a Cartesian product of intervals, parameterized by the upper and lower limit component-wise vectors $\overline{u}$, $\underline{u}$: $U \subseteq \{u \in \mathbb{R}^m \mid \underline{u}_j \leq u_j \leq \overline{u}_j\}$. With a small change in analysis, an input ellipse (or any scaled norm ball) can be considered.

The aforementioned assumptions can be summarized as follows:

Assumption 1 *The dynamics F are twice differentiable, the reward R is state-dependent and Lipschitz, and the input set U is bounded in a product of intervals.*

$$F \in C^2(X \times U; X) \qquad R \in \mathrm{Lip}_1(X; \mathbb{R}) \qquad U \subseteq \{u \in \mathbb{R}^m \mid \underline{u}_j \leq u_j \leq \overline{u}_j\}$$

where $\mathrm{Lip}_1(X; \mathbb{R})$ is the space of Lipschitz functions from X to $\mathbb{R}$. As X is compact and R is Lipschitz, R is therefore bounded.

We recall a previous result that discrete, finite MDPs can be solved with the Upper Confidence Bounds for Trees (UCT) algorithm [30]:

Lemma 1 *Consider an MDP $\langle X, U, F, R, D, K, \gamma \rangle$ with a finite input set $|U| = b$. For initial state x_0, UCT running on a decision tree of depth K and branching factor b over $\mathcal{M}$ yields a value estimate, as a function of number of iterations ℓ.*

$$V^*(x_0) - \mathbb{E}[\hat{V}(x_0, \ell)] \leq O\left(\frac{bK \log(\ell) + b^K}{\ell}\right), \tag{2.2}$$

Remark 1 *The UCT result [30] considers randomness in the algorithm, dynamics, and reward. We consider deterministic dynamics and reward, and thus the expectation is only over the randomness in the algorithm as indicated in Line 5.*

Remark 2 *This result does not include the notion of non-allowable states Ω. We can consider a modification of the input set to be state-dependent, $U(x)$, where in the deterministic setting, the only allowable actions are those that send you to an allowable next state:*

$$U(x) = \{u \mid F(x, u) \in X \setminus \Omega\} \tag{2.3}$$

The UCT [19] result uses non-stationary bandit analysis to show that the tree policy's logarithmic exploration guarantees convergence of the value estimate. While a useful tool, this result only applies to MDPs with a finite action set, and additional treatment is required to apply it to the setting of continuous action spaces. Compared to the a naive "discretize-then-search" approach, SETS provides complexity reduction in both the branching factor and depth.

The last preliminary definition is the reachable set and the Hausdorff distance between sets:

Definition 2 *Given an initial state x_0, a dynamical model F, and a set of actions U, the reachable set is the set of states after rolling out each of the actions:*

$$\mathcal{R}_F(x_0, U) = \{F(x_0, u) \mid \forall u \in U\} \tag{2.4}$$

Definition 3 *Consider a point $a \in \mathbb{R}^n$ and a set $M \subset \mathbb{R}^n$. The point-set distance between a and M is $d(a, M) = \inf_{m \in M} \|a - m\|$. Consider two sets $M, N \subset \mathbb{R}^n$. The Hausdorff distance (standard set distance) between M and N is the symmetric function $d_S(M, N) = \max\{\sup_{m \in M} d(m, N),\ \sup_{n \in N} d(M, n)\}$.*

Convergence to bound of global optimal

The bulk of our analysis is showing that SETS abstracts the original MDP into a discrete MDP of bounded equivalent optimal value, which we accomplish in two steps: (i) the optimal value function of any two MDPs with shared reward and dynamics is bounded by the set distance between their reachable sets and (ii) the reachable set induced by SETS has a bounded distance with the reachable set of the original MDP.

We present the first step about the equivalence of two MDPs.

Lemma 2 *Consider two MDPs satisfying the preceding assumptions:*

$$\mathcal{M}_1 = \langle X, U_1, F, R, D, K, \gamma \rangle \tag{2.5}$$

$$\mathcal{M}_2 = \langle X, U_2, F, R, D, K, \gamma \rangle\,, \tag{2.6}$$

Let V_1^* and V_2^* denote the respective optimal value functions of the two problems. The difference in optimal value function between the two problems is uniformly

bounded by a constant times the maximum set distance between their reachable sets:

$$\|V_1^* - V_2^*\|_\infty \leq \frac{L_R + \gamma L_V}{1-\gamma} \max_{x \in X} d_S(\mathcal{R}_F(x, U_1), \mathcal{R}_F(x, U_2)) \tag{2.7}$$

We now state the second step: the reachable set induced by SETS has a bounded distance with the reachable set of the original MDP (allowing us to apply Lemma 2).

Proposition 1 *Consider an MDP $\langle X, U, F, R, D, K, \gamma \rangle$. When performing Spectral Expansion at state x_0 with horizon H, the H-step reachable set $\mathcal{R}_{F^H}(x_0, U^H)$ differs from the reachable set generated by spectral expansion $\mathcal{R}_{F^H}(x_0, U_{\text{SETS}})$ as:*

$$d_S(\mathcal{R}_{F^H}(x_0, U^H), \mathcal{R}_{F^H}(x_0, U_{\text{SETS}}) \leq \frac{L_{\nabla F}\varepsilon^2}{2} \frac{(L_F)^H - 1}{L_F - 1} \tag{2.8}$$

$$+ 2\sigma_{\max}(S) L_F \frac{(L_F)^H - 1}{L_F - 1} + \frac{L_{\nabla F_{cl}}\varepsilon^2}{2} \frac{1-\alpha^H}{1-\alpha} \tag{2.9}$$

for a ε-ball centered at $(x_0, \bar{u})$ containing $\mathcal{R}_{F^H}(x_0, U^H)$. We recall that S is the input rescaling matrix specified in Line 19 of Algorithm 1.

The intuition for this result is supported by the visualization in Fig. 2.2. We seek to show that the collection of discrete points generated by SETS (green dots) cover the original problem's nonlinear reachable set (green continuous). We do this by introducing two intermediate sets: the reachable set of the linear system subject to energy constraints (blue continuous) and the collection of states constructed from the spectrum of the controllability Gramian (blue dots). To compute an upper bound for the desired set distance, we apply the triangle inequality and use the operations of Spectral Expansion to bound each of the terms:

$$\begin{aligned} d_S(\mathcal{R}_{F^H}(x_0, U^H), \mathcal{R}_{F^H}(x_0, U_{\text{SETS}})) \leq \underbrace{d_S(\mathcal{R}_{F^H}(x_0, U^H), \mathcal{R}_{L^H}(x_0, U^H))}_{\text{Taylor's Theorem}} \\ + \underbrace{d_S(\mathcal{R}_{L^H}(x_0, U^H), \{\mathbf{z}_H^i\}_{i=1}^{2n})}_{\text{Linear Analysis}} + \underbrace{d_S(\{\mathbf{z}_H^i\}_{i=1}^{2n}, \{\mathbf{x}_H^i\}_{i=1}^{2n})}_{\text{Contraction-Theoretic Control}} \end{aligned} \tag{2.10}$$

Our main theoretical result follows from applying Lemma 2 and Proposition 1: SETS abstracts the continuous MDP into a bounded equivalent discrete MDP, UCT solves discrete MDPs, and the desired result follows from application of the triangle inequality:

Theorem 1 *Consider an MDP $\langle X, U, F, R, D, K, \gamma \rangle$. For initial state x_0, SETS with horizon H yields a value estimate, as a function of number of iterations ℓ, satisfying:*

$$|V^*(x_0) - \mathbb{E}[\hat{V}(x_0, \ell)]| \lesssim \underbrace{O\left(\frac{(2nK)\log(\ell)}{H\ell} + \frac{(2n)^{\frac{K}{H}}}{\ell}\right)}_{\text{convergence}} + \underbrace{\frac{C_0 + \gamma^H}{1 - \gamma^H}\left(C_1(1 + C_2\Delta t)^H + C_3\right)}_{\text{steady-state error}}, \tag{2.11}$$

where n is the dimension of the state space X and $C_{0,1,2,3}$ are problem-specific constants, independent of H or ℓ.

To unpack the terms in Theorem 1, the first term, labeled *convergence*, is a decreasing function of both the SETS horizon H (which indicates the branch length of the tree search) and the number of iterations ℓ. The first term goes to zero as the number of iterations ℓ increases and an increase of the hyperparameter H improves the convergence speed of the value estimate: a longer branch length results in fewer overall decisions, and therefore a faster convergence. While providing faster convergence, the trade-off of a longer branch length is a larger asymptotic error, shown in the second term, labeled *steady-state error*. We visualize the trade-off in Fig. 2.3 for a motion planning problem, where a 2D double integrator starts at the blue dot on the left and is tasked to reach the green dot on the right. We vary the branch length parameter H and the number of simulations L and each plot is shaded by the value of its trajectory. The empirical trend of the value is predicted by theory: large H trees converge quickly to sub-optimal solutions, and small H trees converge slowly to highly optimal solutions.

The error term can also be controlled with other parameters. For example, the term with the worst growth behavior, $(1 + C_2\Delta t)$, can be made arbitrarily small by decreasing the integration time of the simulator. However, making this change incurs a higher computational cost per trajectory, limiting the number of trajectories ℓ, finished by the end of the allocated planning budget. Similarly, the entire error term can also be controlled by decreasing the discount factor γ, at the cost of using less information about the future.

Our theoretical analysis validates the algorithm and provides an explainable interpretation of the decision making process. In addition, the discussion of the effect of branch length H and, to a lesser extent Δt and γ, makes it clear that our analysis also enables systematic parameter design of various decision making agents.

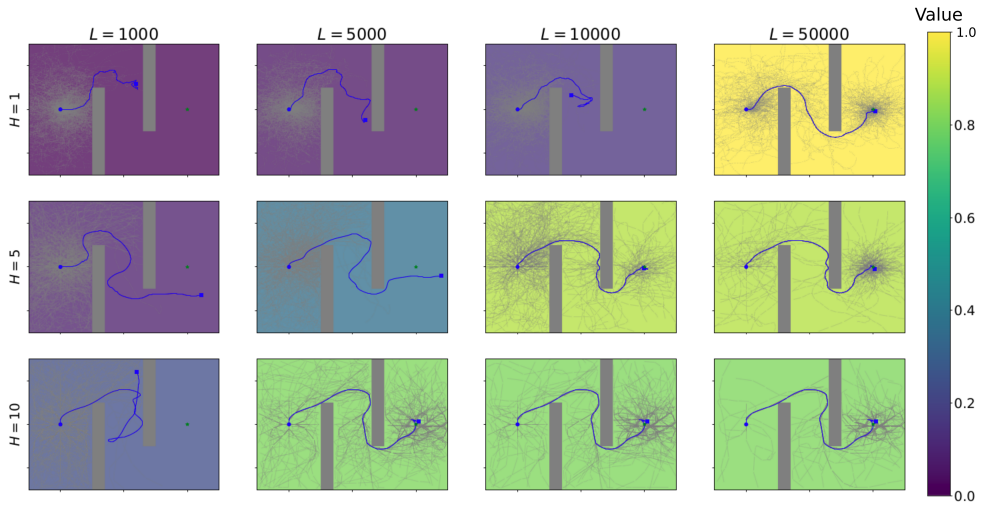


Figure 2.3: Optimal Convergence Rate and Asymptotic Error for Double Integrator System.

For example, if a robot operates in a dynamic environment and has to re-plan frequently to react to new information, the designer can tune parameters to sacrifice some combination of asymptotic error (larger H), dynamics fidelity (larger Δt), or long-term planning (smaller K). Alternatively, when a robot operates in a relatively static environment, but has complex long-term behavior to discover, the designer should allocate more budget to each plan. The former example corresponds to our quadrotor experiment in Sec. 2.5 and the latter example corresponds to the glider experiment in Sec. 2.5.

2.5 Experimental Results

In this section, we present the results of our algorithm in four experiment, three on different robotic platforms and one in simulation. SETS demonstrates new decision making capabilities on a quadrotor in a windy arena, a ground vehicle driving through an obstacle course, and a team of spacecraft catching and redirecting a piece of debris in a fictitious space environment. Our breadth of experiments highlights the ease of deploying SETS to new robotic platforms. We compare against state-of-the-art baselines in simulation, and analyze the empirical results of our theoretical guarantees. The implementation details and videos for all experiments are included in the Supplemental Materials.

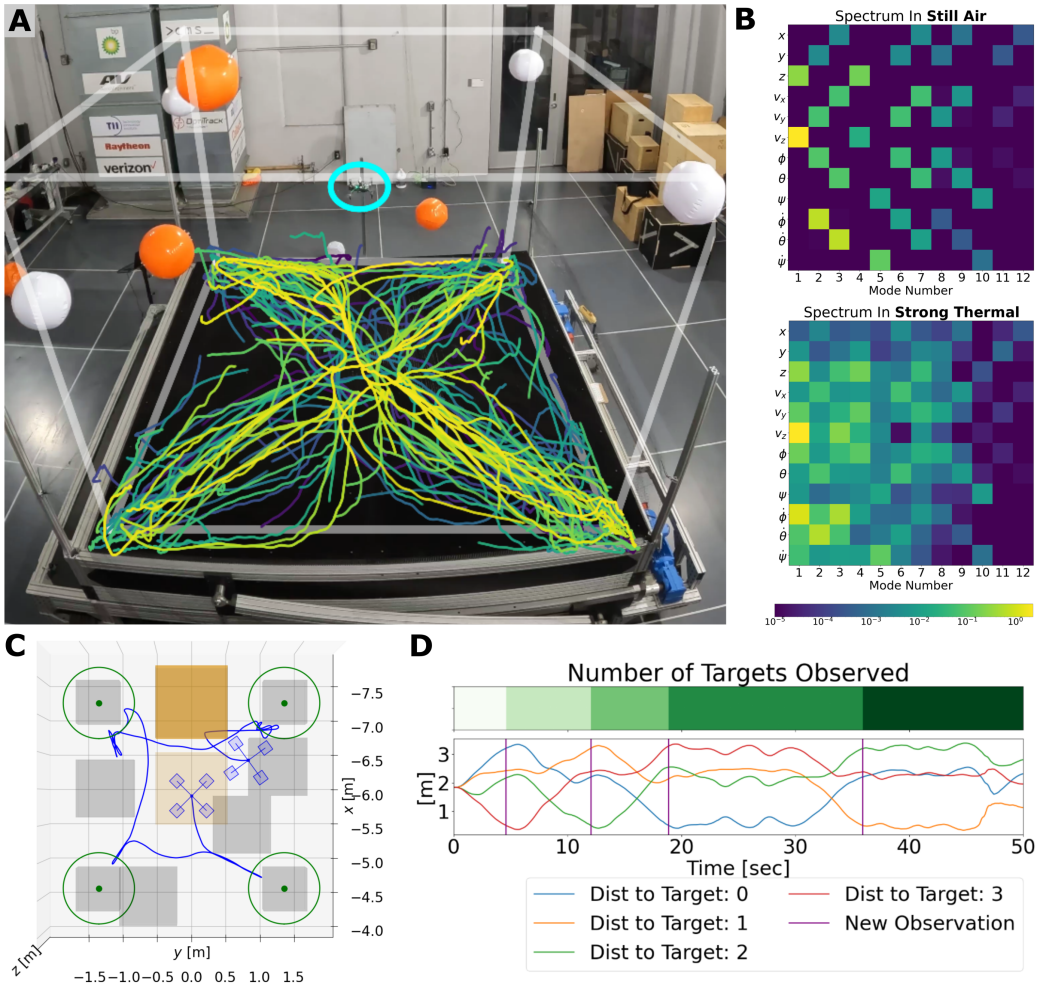


Figure 2.4: Quadrotor Experiment.

Quadrotor navigates a dangerous wind field

Our first experiment uses SETS to plan trajectories for a quadrotor to monitor multiple targets in a cluttered 3D environment of dangerous wind drafts and moving obstacles and is shown in Fig. 2.4.

This experiment is designed to test the ability of SETS to quickly plan in high-dimensional space subject to quadrotor's dynamics and external forcing from aerodynamic interactions. In particular, the dynamics model used by SETS is the standard quadrotor model augmented with a DNN to model the learned residual wind force. In addition, the algorithm must run in real-time to offer corrections to drift and to react to new information.

This problem is challenging for existing solutions because the varying wind strength and its effect on the dynamics determine path feasibility. Therefore, the prob-

lem is not decomposable into position path planning then tracking control. Furthermore, an indicator reward, dense obstacle configuration, and multiple goal regions make it challenging to accurately model with conventional motion planning or optimization-based frameworks. Instead, the algorithm must find a global solution by searching through complex dynamics.

The experimental arena, shown in Fig. 2.4A, is a cube with 3 m side lengths where the Caltech Real Weather Wind Tunnel generates controlled columns of air to suspend and move spherical obstacles and observation targets. In addition to the physical obstacles, there exist dangerous and benign regions of flow of varying speed that affect the planning problem. The search tree is visualized by projecting the 12-dimensional state trajectories onto the 2-dimensional surface of the fan array. The branches are colored by when they were expanded: purple are the first trajectories in the tree and yellow are the last trajectories in the tree. Monte Carlo Tree Search algorithms adaptively concentrate on promising trajectories, and therefore the yellow trajectories serve as an indicator of the plan of the algorithm: we can see that these trajectories concentrate to stretch between the origin and each of the target locations.

The SETS tree is constructed with the spectrum of the locally linearized controllability Gramian, shown in Fig. 2.4B. The modes in still air are intuitive: the first mode (column) of the spectrum corresponds to accelerating the vertical velocity, v_z, and the next two modes correspond to a pitch and roll maneuver, ϕ and θ, respectively. The modes in the thermal are more diffuse and are beyond human intuition, yet they are a provably efficient representation of the complex dynamics. For example, it becomes much harder for the quadrotor to excite the yaw, ψ, degree of freedom independently from the other degrees of freedom.

The resulting trajectory from this experiment is shown in Fig. 2.4C, where we see the quadrotor visit all the targets while avoiding obstacles and dangerous winds. In order to observe the target, the quadrotor has to be within a sensing radius of 50 cm of the center of the target (visualized in Fig. 2.4B). The physical size of the target creates an obstacle of 30 cm radius, leaving a feasible viewing volume of 0.3 m^3 for each target. The ability to find and stitch these “needle-in-a-haystack" pieces of the solution while executing a precise and dynamically stable maneuver demonstrates the precision of SETS’s exploration.

The overall mission progress is described the by the the distance to each target over time is plotted and the cumulative number of targets observed, as shown in

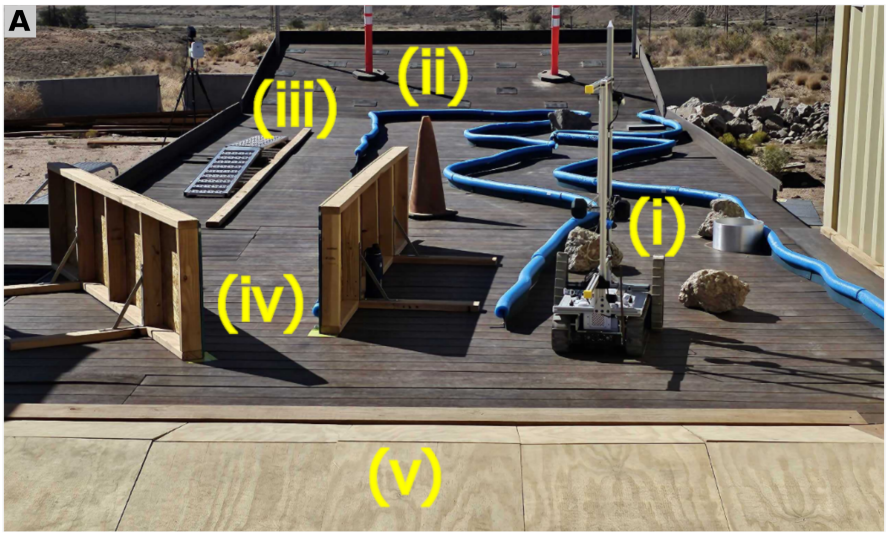
Fig. 2.4D. The quadrotor visits all the targets in 37 seconds, with the final transition through the narrow corridor of thermals taking the longest time to find a solution. The solution's nature is reminiscent of the Traveling Salesman Problem [31], where SETS discovers the cost between targets and the optimal path to visit them all. Discovering this solution automatically, rather than prescribing a sequence of waypoints, has two advantages: first, the burden on the designer to enumerate and integrate many behaviors is relieved, extending the operational envelope of the robot and second, SETS may solve problems beyond the intuition of the designer. For example, it is not clear to a designer at what fan strength and size a wind gust becomes too dangerous to cross safely.

Tracked vehicle shares autonomy

Our second experiment, shown in Fig. 2.5, uses SETS for driver assist of a tracked vehicle subject to antagonistic terrain effects, tipping constraints, obstacle avoidance, and actuator degradation. This human-in-the-loop collaboration is also called "parallel autonomy" in the self-driving car literature [13]. These experiments were developed during the DARPA Learning Introspective Control (LINC) research program, with various versions of the algorithm tested at the Sandia National Laboratory Robotic Vehicle Range. A video of this experiment is presented in Movies 2 and 3.

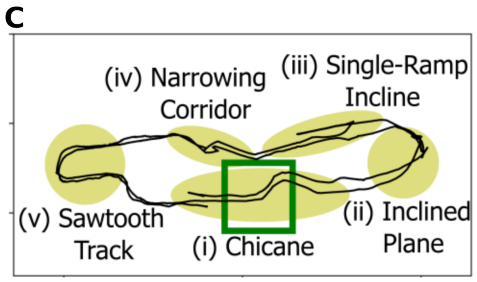
In the experiment, the tracked vehicle is tasked to assist a driver to traverse a 20 m $\times$ 10 m test circuit with slippery slopes, a variety of obstacles, and a thin chicane track, shown in Fig. 2.5A/C. In Fig. 2.5D, we zoom in on one section of the chicane track, in which our algorithm automatically adjusts the driver's command to maintain safety. Passing through a particularly narrow portion of the track (less than 5 cm clearance), without the need for an updated command from the driver, the robot slows and turns to avoid hitting the walls from a nominal speed of 1 m/s to 0.25 m/s. SETS predicts a collision if no diversionary maneuver is taken, then creates and executes an updated plan.

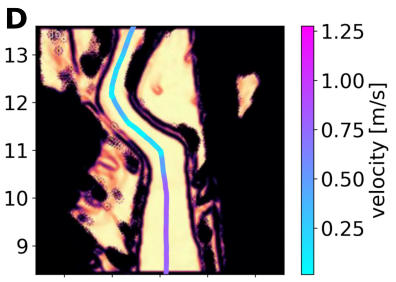
This problem is difficult to solve with existing methods because in our experiments, human pilots prefer an interface that tracks commanded speed, rather than commanded waypoints, meaning a position-space goal region for motion planning does not capture the human-robot interaction well. In addition, optimization-based approaches are challenging because exploration is needed to consider nearby candidate trajectories to assist the driver. For example, the human driver may command



B

Track Section	Chicane Track	Inclined Plane	Single-Ramp Incline	Narrowing Corridor	Sawtooth Track	Total
Safety Violation Count (driver alone)	2	0	1	3	0	6
Safety Violation Count (with SETS)	**0**	0	**0**	**0***	0	**0**





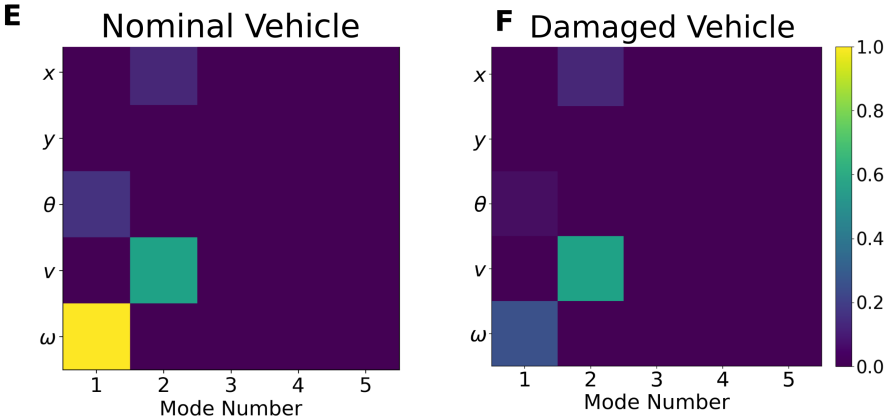


Figure 2.5: Tracked Vehicle Experiment

a forward velocity, not understanding there exists an impending collision or risk of tipping over, and instead of coming to a halt, our method explores alternative plans including slowing down, backing up, and turning that ultimately provide the commanded velocity tracking and match the operator intention. SETS adjusts the pilot's command to maintain safety while maintaining forward progress, displaying intricate maneuvers such as navigating closely around obstacles, decelerating when traversing ridges to avoid tip-over, and executing reverse turning maneuvers in tight corners, all while experiencing adversarial track degradation and time-varying dynamics.

While moving through the track, the vehicle is subject to actuator degradation that scales the control limits of each drive motor by 25% in an alternating sequence of separate degradation and mixed degradation. This attack signal is randomly toggled on and off with a period of approximately ten seconds. SETS is able to efficiently interpret this attack signal through the local spectrum of the tracked vehicle, shown in Fig. 2.5E/F. Here, the locally linearized dynamics have a two-dimensional controllable subspace, one dimension associated with moving forward/backward and one dimension associated with turning left/right. First, we note the effect of degrading the actuator on the spectrum: the controllability of turning mode is decreased, providing the tree search an expressive interpretation of the current physical situation. Second, we note that the mode numbers 3, 4, and 5 have zero magnitude, implying the tracked vehicle is not locally controllable [32]. Our method provides an informal notion of nonlinear global controllability for systems that are not locally controllable. By stitching locally linear trajectories, SETS creates motions that are not available to the linearized systems on their own. For example, a sideways translation can be achieved through the sequential combination of a forward snaking motion followed by a backward snaking motion, reminiscent to motion planning strategies in literature [33].

In program demonstrations, the same professional driver drove through the track with and without an autonomy driver assist. Throughout the experiment, the expert driver was asked to command the vehicle in the same adversarial manner with intentional commands to try to force a safety error. We show the number of safety violations in a selected run of the test circuit where adversarial and changing disturbances are present in Fig 2.5B, where we see SETS outperformed the driver alone and completely prevented safety violations. In the chicane section, at 2:36 in Movie 1, SETS causes the robot to turn and avoid a collision with the track, despite a driver

command that would cause an impact. In the chicane section, at 2:45 in Movie 1, SETS causes the robot to act more conservative, slowing down as it traverses the narrow section. In addition, during program demonstrations, using SETS, even an inexperienced driver is able to safely navigate the course.

SETS acts as a planning module which interacts with custom perception, control, and safety algorithms. Running in real-time model-predictive control fashion, SETS runs at 10 Hz, generating trajectories 1.6 seconds into the future. SETS's ability to solve general MDPs facilitates its interaction with the other autonomy components. For example, the hazard map combines foundational vision models for segmentation with geometric and dynamic information to predict traversability, enabling complex behavior: in one case, the vehicle is on a narrow path with a large rock and a brick blocking the way. Using information from the hazard map, SETS is able to plan a safe trajectory, rolling over the brick while avoiding the rock. The problem is infeasible if the algorithm naively considered both obstacles as impassable. Only by integrating the unstructured information of the hazard map, is SETS able to generate a safe plan. SETS also interacts with an adaptive controller, which compensates for terrain changes and actuator degradation using parameter adaptation to rapidly update the corresponding dynamic function. SETS uses the system identification of the adaptive controller to update the dynamics function of the MDP. The real-time nature of SETS and its ability to plan over a wide class of dynamics allow it to incorporate nonlinear dynamics updates and benefit from this real-time interaction.

Spacecraft team redirects debris

In the third and final hardware experiment, we deploy SETS on a team of spacecraft to capture and redirect a piece of space debris with a tether, reminiscent of a NASA mission concept to capture and redirect a near-Earth asteroid [34]. This experiment tests the ability of SETS to coordinate multiple agents and plan through high-dimensional nonlinear dynamics induced from contact forces and tether dynamics.

The experiment, depicted in Fig. 2.6A, takes place in an arena with a smooth and flat acrylic floor, where spacecraft robots hover on air bearings and are actuated with onboard thrusters to simulate a frictionless space environment. SETS controls the two cooperative and tethered spacecraft, and the third spacecraft is used to model an uncooperative target that does not use its thrusters. The cooperative spacecraft are

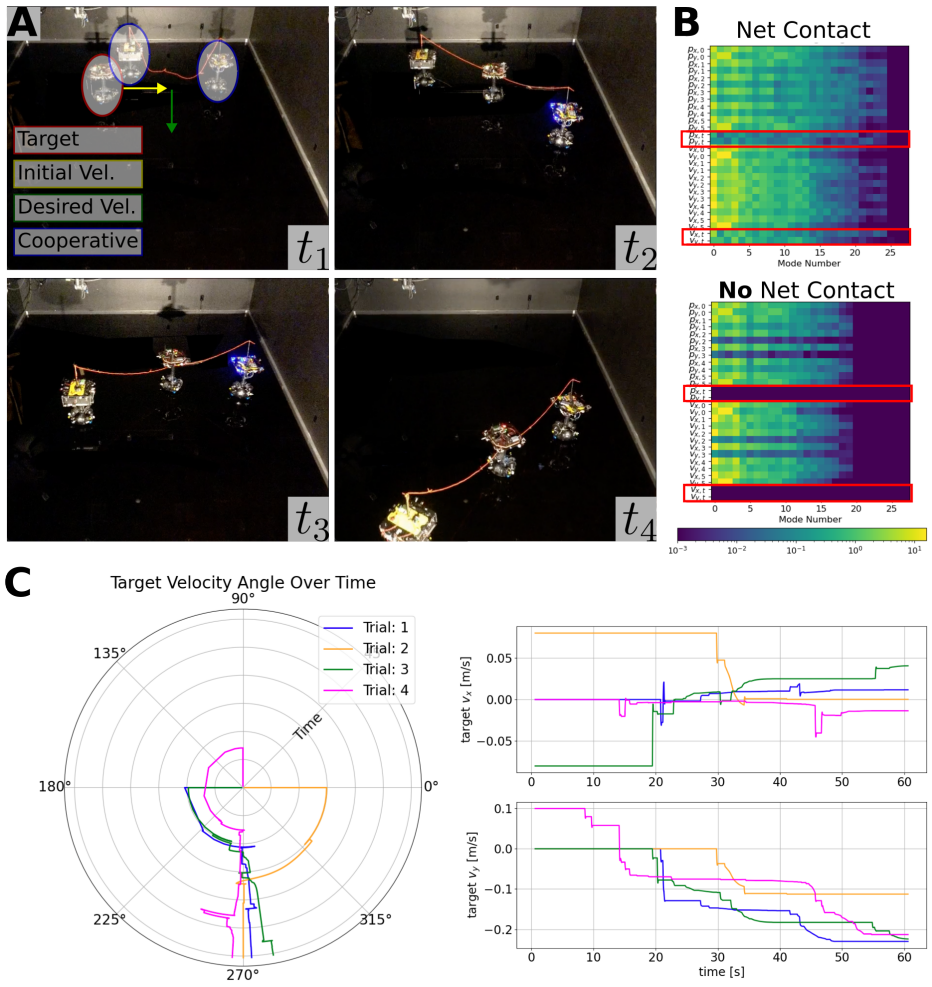


Figure 2.6: Spacecraft Experiment.

tasked to arrest the motion of the target and shepherd it to exit the arena in a desired direction. SETS predicts the motion of the three bodies and the tether, which is modeled with a spring-mass-damper finite-element-model, while only controlling the thruster inputs of the two cooperative spacecraft. This high-dimensional and underactuated problem tests the ability of SETS to plan for complex dynamics over a horizon. In particular, the two controlled spacecraft can only affect the dynamics of the target through contact with the center of the net, and the center of the net can only be influenced by the thrusters after propagating through the lattice elements of the tether. This chain of dynamics requires deliberate maneuvering of the controller spacecraft to redirect the target.

Despite these challenges, SETS automatically generates an efficient representation: in Fig. 2.6B, the controllable modes of the system capture this networked structure

of the finite element tether model. The most controllable states are those associated with the outermost nodes in the network, which are exactly the controlled spacecraft. The controllability of the nodes in the net decreases towards the center of the structure. Importantly, the spectrum reveals that actuating the target is impossible before contact.

SETS interprets this discrete representation of the dynamics to efficiently find near-optimal solutions. Four trials are tested by varying the target's initial position and velocity. In the first, the target spacecraft is stationary and SETS finds a trajectory that sequentially deploys, captures, and redirects. In the second and third configurations, the target is initialized moving parallel to the initial cooperative spacecraft configuration, and quick motion is required of the tethered spacecraft to move into its way for capture. In the fourth, the target is initialized moving towards the initial cooperative spacecraft configuration, and they execute a "trampoline-like" maneuver, pulling the tether taut to springboard the target back in the desired direction.

The trajectory data for each case are shown in Fig. 2.6C. The polar plot shows the relative angle between the x-and y-components of the target's velocity over time, and the two standard plots show the absolute velocities over time. The polar plot, which is coordinate-aligned with the snapshots in Fig. 2.6A, shows the relative angle for each trial converge to 270°, which is the desired mission behavior. Successfully controlling this high-dimensional and highly underactuated system in real-time is only possible with SETS and ultimately enables an important new mission concept.

Aerodynamic glider performs persistent observation

In this numerical experiment, we use SETS a six degree-of-freedom glider is tasked with target observation in the presence of a thermal updraft shown in Fig. 2.7. We analyze tree data to make observations about the policy and value convergence and select the branch parameter H in a principled way. We also use this experiment to compare against external baselines and make ablation studies.

The environment is a $2 \times 2 \times 1$ km^3 volume arena and takes place over 10 minutes in simulation time, shown in Fig. 2.7A. The glider dynamics, parameters, and aerodynamic coefficients are from the Aerosonde UAV [35]. The interaction of aerodynamics, the observation objective, and the environmental thermal makes this an interesting planning problem. In order to generate enough lift to counteract gravity, the glider needs to average approximately 30 meters per second forward

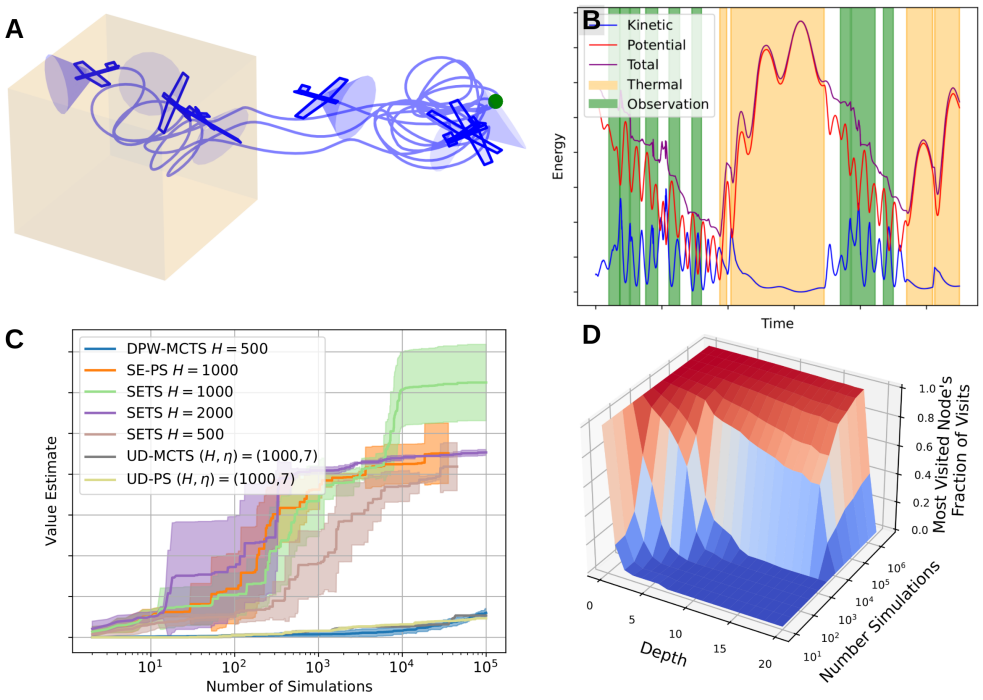


Figure 2.7: Glider Experiment.

velocity. At the same time, drag drains the system's kinetic energy and, without exploiting the thermal, would cause the glider to crash into the ground after about 4 minutes. However, the glider can save itself by flying into the thermal, extracting energy from the environment, and resuming its observation task. SETS discovers this solution in real-time, running in model-predictive control fashion by generating 100 seconds of trajectory in every 45 seconds of real time on a standard laptop processor. The trade-off between kinetic and potential energy in this periodic solution are visualized in Fig. 2.7B. This periodic solution is challenging to generate with existing motion planning or convex-optimization frameworks because a path to a static goal region would either eventually cause the glider to crash or never observe the target, and the detour to the thermal is in contradiction of the smoothed reward gradient to the goal.

The data for the two subfigures, Fig. 2.7C/D, are generated by running SETS from a given state, and analyzing the tree data. In Fig. 2.7C, we plot SETS's root node value estimate versus the number of simulations. As predicted by our theoretical result, Theorem 1, the branch length parameter controls the trade-off between convergence rate and error: short branches have slow convergence to small steady-state error, and long branches have fast convergence to large steady-state error. The H =

500 case induces such a high complexity problem that it is unable to overtake the $H = 1000$ estimate within the operating regime of 10^5 simulations (10 minutes of wall-clock time). This plot informs our decision to select the optimal branch length $H = 1000$.

We use the same value estimate data in the baseline study, which is designed to isolate the effect of representation and exploration strategy. For choices of representation (low-level construction of nodes and edges), we implement spectral expansion (SE), uniform discretization (UD), and double progressive widening (DPW) [36]. The SE method is ours, the UD approach discretizes the action space with η discrete points per dimension, and the DPW method samples from the action space where more samples are given to promising nodes. For choices of exploration strategy (high-level search on existing nodes and edges), we implement Monte Carlo Tree Search (MCTS) [19] and predictive sampling (PS) [37], where PS is a uniform sampling approach that returns the best trajectory found. The baselines and our method in Fig. 2.7C are a straightforward permutation of representation and exploration strategy. Finally, the baseline data in Fig. 2.7C is generated by running each solver over 10 random seeds for branch lengths $H = 100, 500, 1000, 2000$ and discretization levels $\eta = 3, 5, 7, 11$, where only the best variant of each baseline is shown. Other than our approach, the only algorithm that performs well is spectral expansion with predictive sampling, (SE-PS). In fact, SE-PS $H = 1000$ outperforms SETS $H = 500$, suggesting that representation, not exploration strategy, is the most important component of optimal decision making for dynamical systems.

In Fig. 2.7D, we define a measure of the tree's confidence at a particular depth as the most visited node's fraction of the total visit count, and plot it versus the depth and size of the tree. The relative visitation frequency is a metric for the tree's confidence in its action selection because of the Upper Confidence Bound rule of Monte Carlo Tree Search [19]: actions are selected by their optimistic estimated value and when the visitations are highly concentrated to a single action, the optimistic estimated value of the other actions is less than that of the highly visited (and therefore well-estimated) selected action. As the number of simulation increases, concentration occurs deeper in the tree, indicating the plan is being refined further into the future, and that the tree has reached sufficient confidence in the plan until that point. When running in receding-horizon fashion, the time between replanning steps should allow a highly confident plan to develop.

2.6 Related Work

We present a qualitative comparison with existing approaches and provide perspectives on how our work will impact the future of robotics.

Optimization and gradient-based planning

Some decision making problems can be solved with convex optimization techniques [6, 7]. Taking advantage of fast numerical solvers (e.g. [38]), this family of methods has been shown to solve, in real-time, a variety of challenging dynamical problems including quadrotor drone racing [39], in-orbit assembly[40], bipedal locomotion [41], swarm coordination [42] and swarm coverage [43]. A variety of techniques, including sequential convex programming [6], collocation [41], and single and multiple shooting methods [44–46] enable this method to be applied to non-convex problems, though theoretical analysis of these methods has shown convergence is limited to a local minima [47, 48]. However, the general decision making problem's state, input, dynamics, and reward specifications can each create local minima traps under which a pure exploitation strategy will fail to find the global optimal solution. The classical example is the bug trap problem [28], where the obstacle configuration will cause methods that do not explore to converge to a highly sub-optimal point. Recent inspired work [49] has proposed avoiding local minima in motion planning by partitioning the environment into convex sets and solving a relaxed mixed-integer program.

Compared to these approaches, SETS provides globally optimal solutions by performing a discrete search on a carefully constructed representation. For example, the quadrotor navigates around local minima traps from obstacles and wind gusts and the glider temporarily ignores the high reward of visiting the target and instead navigates to the thermal to maintain altitude and energy constraints.

Sampling-based motion planning

The standard robotics approach to overcome the nonconvexities inherent in problem data, such as traps from obstacle configurations, is sampling-based search [5]. Foundational methods such as Probabilistic Roadmaps (PRM) [4], Rapidly-Exploring Random Trees (RRT) [3] and their variations (e.g. RRT* [50]) sample configurations and use a local planner to build trees and graphs on which to perform global optimization. Although RRT was originally designed for kinodynamic planning, these planners are most commonly used as geometric path planners.

Applying sampling-based planning for dynamical systems has two challenges: first,

the higher-order states increase the dimensionality of the system, resulting in slow convergence and poor performance and second, these methods rely on the existence of a local planner capable of solving arbitrary two-point boundary value problems. Addressing these challenges is an active area of research. For example, Stable-Sparse-RRT (SST) [51] relaxes the knowledge of a local planner by sampling control inputs to generate dynamically feasible paths, then prunes redundant edges to maintain a sparse tree. Another work [52] considers quasi-Monte Carlo sampling to speed convergence of tree search for control-affine and driftless control-affine systems. Discontinuity-bounded A* [53] is similar to our work in that they plan for high-dimensional dynamical systems over a discrete structure of motion primitives, and is different because their motion primitives are generated offline by sampling input and goal states. The generation and integration of motion primitives into discrete planning has a rich history [54–56].

As in our work, the Differential Fast Marching Tree method [57] mentions the controllability Gramian for planning. Whereas this work considers two-point boundary value problems for linear systems with drift, and uses the Gramian to check reachability from states sampled from the high-dimensional space, we consider a broad class of nonlinear systems and non-convex rewards, and avoid sampling the high-dimensional space by using the Gramian's spectrum to direct exploration. Other inspired works that blend control and planning are LQR-trees [58] and funnel library planning [59], both of which rely on sum-of-square programming to compute regions of attraction upon which the discrete planner searches.

The search strategy of SETS is unique in that it does not sample from the state or control space, a strategy which scales poorly in high-dimensional spaces, and instead rely on the spectrum of the locally linearized dynamics to automatically construct motion primitives in real-time. This conceptual difference manifests itself in new robotic capabilities: to the best of our knowledge, there do not exist kinodynamic motion planners that *search* in real-time through a high-dimensional nonlinear dynamics such as a 12-dimensional quadrotor with DNN-modeled wind effects.

In addition to the differences in search strategies, our work differentiates itself in the generality of the problem assumptions. Whereas the conventional motion planning problem requires a static, position-space goal region [60], the decision making problem relaxes this assumption. This is useful in instances where a goal region does not capture the essence of the problem, or when the goal region is difficult

or impossible to define. For example, in chess, a notion of a goal region could be "the set of all positions in which the opponent king is checkmated", but this set is challenging to enumerate or path to with a motion planning method. With this in mind, the experiments in this paper are selected because they are easily formulated as decision making problems, but they cannot be cast as motion planning. For example, in the tracked vehicle experiments, human drivers preferred interfacing with an algorithm that tracks their commanded velocity rather than navigates to a waypoint, implying a conventional motion planning framework with a position goal region does not model the human-robot interaction well. The minimally invasive velocity-tracking interaction is similar to that presented in [61]. In the glider experiment, the glider is not tasked to plan between the thermal and target, but to simply find the best way to observe the target. A static goal region does not exist because the optimal behavior is to oscillate between observing the target and the thermal. Besides immediately growing the set of feasible robot behaviors, the more general problem framework places decision making as a more natural foundation to extend to stochastic dynamics [23], partially observable [62–64], and game-theoretic [65] settings.

There is a recent body of work on sampling-based model predictive control methods, including cross-entropy motion planning [66] and model predictive path integral control [67]. By sampling many trajectories then performing a weighted average, these methods have shown impressive performance in scenarios including autonomous driving [67] and manipulation [68]. These methods have also shown straightforward integration with high-fidelity simulators [69] and learned dynamics models [67]. These strategies traditionally perturb inputs with Gaussian noise to promote exploration, but without careful tuning of noise distributions and averaging temperature, this can lead to slow exploration or the generated policies getting stuck in local minima. Alternative sampling strategies have been investigated, such as sampling splines in state space [68], but their theoretical role is unclear. In contrast, our method uses MCTS to automatically balance exploration and exploitation with an established theoretical mechanism based on minimum regret online learning [30], ultimately enabling our algorithm to avoid local minima and guarantee global convergence.

Sampling-based search is also investigated in the partially observable case. Algorithms for handling continuous state and action spaces have focused on particle filtering methods, with specializations to overcome the challenges inherent in un-

certain observations such as sharing observation data in the tree [70], progressive widening [63], or linear filtering [64]. We bring our attention to the fully observed case to focus on the difficulty of decision making, rather than combined decision making and information gathering. There is recent work that transforms a stochastic planning problem into a deterministic planning problem [71], moving the difficulty of dealing with stochastic dynamics and observations into same decision making framework we consider.

Reinforcement learning

The decision making problem in its full generality is studied in the reinforcement learning community. The most comparable technology in this field is the planning component of model-based reinforcement learning; once the dynamics and reward model are learned, the planning component finds the optimal value and policy. The classical planning methods using dynamics and reward models are value and policy iteration [2, 8]. The fundamental issue with these methods, which persists in modern approaches, is that representing a high-dimensional continuous space has a high complexity [72, 73]. The direct approach is to discretize the state and action space and run value iteration, but this has a storage complexity that is exponential in the state dimension, making these methods computationally infeasible for high-dimensional robot applications. Research has developed in multi-grid [74] and adaptive-grid [75] representations, but the practical gains in scaling to high-dimensional systems are marginal. Recent inspired work [76] has proposed a tensor-based method that exploits a low-rank decomposition of value functions, enabling efficient discretization and improved policy generation on systems of comparable dimensionality to our experiments.

An alternate reinforcement learning approach is that of model-free methods that directly learn correlations between observations and optimal actions without explicitly using a dynamics or reward function [77, 78]. Policy gradient methods have offered a powerful technique to handle continuous state and action spaces, with methods such as Proximal Policy Optimization [79] becoming a standard tool in the community. In special cases, such as linear-quadratic regulation [80], global convergence results exist. However, the general convergence of these algorithms in continuous space is limited to local stationary points [81]. Additional work has gone on to classify these stationary points between problems, drawing conclusions about structural similarities that help policy gradient methods converge globally [82]. These methods, while powerful and general, require large datasets

and an offline training phase. These methods suffer fundamentally from *domain shift*, the difference between the offline training environments and the environment of the deployed system. In contrast to data-driven reinforcement learning methods, our algorithm is able to run on a never-before-seen problem with guaranteed global convergence to an optimal solution, thus avoiding the danger of domain shift.

A natural direction to develop real-time intelligence is via anytime algorithms that can be stopped at an arbitrary point, returning satisfactory solutions that increase in quality as more time is given. In the world of decision making, the prototypical example is Monte Carlo Tree Search [21], an algorithm that simulates random trajectories while biasing towards actions of high reward. Exhaustive [83] and uniformly random searches [37] have been shown to be effective in some scenarios, but the improved strategic exploration of MCTS enables convergence in problems where simpler techniques fail to efficiently search the combinatorially large space. Although MCTS can perform very well in traditional artificial intelligence settings such as games, the complexity of the high-dimensional and continuous world presents a fundamental challenge. Directly sampling the action space [23], even with sophisticated strategies [84, 85], induces trees with large width (sampling high-dimensional continuous action space, creating a high branching factor) and large depth (discretizing time and making a large number of decisions over the horizon).

Temporal abstraction, or options [86], is a principled framework to make decisions over sequences of actions and reduce the complexity of decision making. Options in decision making are analogous to motion primitives in motion planning. There has been work in option construction [87, 88] by hand [89] and by using previously generated solutions to speed up an online evaluation and in option construction [90] in the partially observable setting. In contrast, our solution automatically generates options in a provably correct and widely applicable framework.

Data-driven methods [91], and combinations with gradient-based techniques [92–94], have also been deployed for decision making. However, their reliance on large amounts of demonstration data during an offline training phase limits their applicability to systems and scenarios where the complete problem data is known ahead of time. In contrast, our method can be deployed on a never-before-seen problem, and, for any allowed computational budget, produce a plan of approximately optimal decisions that increases in quality with more time.

2.7 Discussion

We expect SETS to positively impact design of autonomous systems and fundamental research in decision making. From a design perspective, SETS provides many important advantages: first, SETS interfaces naturally with other autonomy components, as demonstrated in Sec. 2.5, and can be used for new and diverse tasks and systems, relieving the burden on the designer and extending the operational envelope of an autonomous robot. Second, because the search tree of SETS can be visualized and analyzed, it has a high degree of explainability and can be tuned and verified by the user. Third, because SETS is efficient enough to run in real-time, it reacts to new information on-the-fly. For these reasons, we believe SETS can be a default choice for planners in a wide range of autonomy applications.

From a research perspective, SETS builds an important connection between dynamical systems and machine learning. To develop this connection, we can interpret the tree policy as an online learning process of a categorical distribution on each node's children. Our theoretical result, which applies to a broad class of dynamical systems, suggests the spectrum of the local controllability Gramian can be used as proveably correct and widely applicable learning features. These features enable real-time learning for complex dynamical systems by simplifying the decision making problem to selecting among a set of natural motions of the system. Although beyond the scope of this work, we foresee the application of these features for offline policy learning, kinodynamic motion planning, and other sampling-based planning [85], providing reduction in computational complexity and improved convergence rates.

From the algorithmic complexity perspective, SETS reduces complexity in two separate mechanisms as compared to the uniform-discretization approach: First, as SETS plans over trajectories instead of individual actions, the total tree depth is decreased by a factor of H, where H is the duration of a trajectory generated by the spectral nodal expansion. Second, the width of the tree (branching factor) is decreased from an exponential dependency on control dimension to a linear dependency on state dimension. This is enabled because each child node uniquely tracks one of the basis vectors of the controllable subpsace and, from linear theory, the number of basis vectors is upper bounded by the state dimension. In contrast, for a uniform discretization, the number of elements in the m-cube has an exponential dependence on the m. The number of combinations in the tree, (i.e. width to the power of depth) is important because it appears in the convergence rate analysis of

MCTS.

BIBLIOGRAPHY

[1] Benjamin Rivière*, John Lathrop*, and Soon-Jo Chung. "Monte Carlo Tree Seach for Dynamical Systems with Spectral Expansion". In: *(Review at Science Robotics)* (2024).

[2] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[3] Steven LaValle. "Rapidly-exploring random trees: A new tool for path planning". In: *Research Report 9811* (1998).

[4] Lydia E Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[5] Andreas Orthey, Constantinos Chamzas, and Lydia E Kavraki. "Sampling-Based Motion Planning: A Comparative Review". In: *Annual Review of Control, Robotics, and Autonomous Systems* 7 (2023).

[6] Daniel Morgan, Soon-Jo Chung, and Fred Y Hadaegh. "Model predictive control of swarms of spacecraft using sequential convex programming". In: *Journal of Guidance, Control, and Dynamics* 37.6 (2014), pp. 1725–1740.

[7] Danylo Malyuta et al. "Convex optimization for trajectory generation". In: *arXiv preprint arXiv:2106.09125* (2021).

[8] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[9] Yashwanth Kumar Nakka et al. "Information-Based Guidance and Control Architecture for Multi-Spacecraft On-Orbit Inspection". In: *Journal of Guidance, Control, and Dynamics* 45.7 (2022), pp. 1184–1201. DOI: 10.2514/1.G006278.

[10] Leslie Pack Kaelbling and Tomás Lozano-Pérez. "Hierarchical task and motion planning in the now". In: *IEEE International Conference on Robotics and Automation*. 2011, pp. 1470–1477.

[11] Caelan Reed Garrett et al. "Integrated task and motion planning". In: *Annual review of control, robotics, and autonomous systems* 4 (2021), pp. 265–293.

[12] Brian Paden et al. "A survey of motion planning and control techniques for self-driving urban vehicles". In: *IEEE Transactions on intelligent vehicles* 1.1 (2016), pp. 33–55.

[13] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. "Planning and decision-making for autonomous vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 187–210.

[14] Yunlong Song et al. “Reaching the limit in autonomous racing: Optimal control versus reinforcement learning”. In: *Science Robotics* 8.82 (2023), eadg1462. DOI: 10.1126/scirobotics.adg1462.

[15] Pieter Abbeel et al. “An Application of Reinforcement Learning to Aerobatic Helicopter Flight”. In: *Advances in Neural Information Processing Systems*. Ed. by B. Schölkopf, J. Platt, and T. Hoffman. Vol. 19. MIT Press, 2006. URL: https://proceedings.neurips.cc/paper_files/paper/2006/file/98c39996bf1543e974747a2549b3107c-Paper.pdf.

[16] Ian Lenz, Honglak Lee, and Ashutosh Saxena. “Deep Learning for Detecting Robotic Grasps”. In: *Proceedings of Robotics: Science and Systems*. Berlin, Germany, June 2013. DOI: 10.15607/RSS.2013.IX.012.

[17] Guillermo A Castillo et al. “Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2021, pp. 5136–5143.

[18] Michael Kearns, Yishay Mansour, and Andrew Y Ng. “A sparse sampling algorithm for near-optimal planning in large Markov decision processes”. In: *Machine learning* 49 (2002), pp. 193–208.

[19] Levente Kocsis and Csaba Szepesvári. “Bandit based monte-carlo planning”. In: *European Conference on Machine Learning*. Springer. 2006, pp. 282–293.

[20] Rémi Munos et al. “From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning”. In: *Foundations and Trends® in Machine Learning* 7.1 (2014), pp. 1–129.

[21] Cameron B Browne et al. “A survey of monte carlo tree search methods”. In: *IEEE Transactions on Computational Intelligence and AI in games* 4.1 (2012), pp. 1–43.

[22] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[23] David Auger, Adrien Couetoux, and Olivier Teytaud. “Continuous upper confidence trees with polynomial exploration–consistency”. In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic*. Springer. 2013, pp. 194–209.

[24] Devavrat Shah, Qiaomin Xie, and Zhi Xu. “Non-asymptotic analysis of monte carlo tree search”. In: *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*. 2020, pp. 31–32.

[25] David Silver et al. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359. URL: https://doi.org/10.1038/nature24270.

[26] Stephen Boyd et al. *Linear Matrix Inequalities in System and Control Theory*. SIAM, 1994.

[27] Kemin Zhou and John Comstock Doyle. *Essentials of robust control*. Vol. 104. Prentice hall Upper Saddle River, NJ, 1998.

[28] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. "The Open Motion Planning Library". In: *IEEE Robotics & Automation Magazine* 19.4 (Dec. 2012). https://ompl.kavrakilab.org, pp. 72–82. DOI: 10.1109/MRA.2012.2205651.

[29] Benjamin Riviere et al. "Neural tree expansion for multi-robot planning in non-cooperative environments". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 6868–6875.

[30] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. "Improved monte-carlo search". In: *Univ. Tartu, Estonia, Tech. Rep* 1 (2006), pp. 1–22.

[31] Merrill M Flood. "The traveling-salesman problem". In: *Operations research* 4.1 (1956), pp. 61–75.

[32] Karl Johan Astrom and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. USA: Princeton University Press, 2008. ISBN: 0691135762.

[33] Richard Martin Murray. *Robotic Control and Nonholonomic Motion Planning*. University of California, Berkeley, 1991.

[34] *Asteroid Redirect Mission Reference Concept*. https://www.nasa.gov/wp-content/uploads/2015/04/asteroid_redirect_mission_reference_concept_description_tagged.pdf. Accessed: 2024-05-16.

[35] Randal W Beard and Timothy W McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton university press, 2012.

[36] Adrien Couëtoux et al. "Continuous upper confidence trees". In: *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*. Springer. 2011, pp. 433–445.

[37] Taylor Howell et al. "Predictive sampling: Real-time behaviour synthesis with mujoco". In: *arXiv preprint arXiv:2212.00541* (2022).

[38] MOSEK ApS. *MOSEK Fusion for C++ 10.1.21*. 2019. URL: https://docs.mosek.com/latest/cxxfusion/index.html.

[39] Philipp Foehn, Angel Romero, and Davide Scaramuzza. "Time-optimal planning for quadrotor waypoint flight". In: *Science Robotics* 6.56 (2021), eabh1221.

[40] Rebecca C Foust et al. "Autonomous in-orbit satellite assembly from a modular heterogeneous swarm". In: *Acta Astronautica* 169 (2020), pp. 191–205.

[41] Ayonga Hereid et al. “3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1447–1454.

[42] Daniel Morgan et al. “Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming”. In: *The International Journal of Robotics Research* 35.10 (2016), pp. 1261–1285.

[43] Mac Schwager, Daniela Rus, and Jean-Jacques Slotine. “Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment”. In: *The International Journal of Robotics Research* 30.3 (2011), pp. 371–383.

[44] David Mayne. “A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems”. In: *International Journal of Control* 3.1 (1966), pp. 85–95.

[45] HG Bock et al. “A direct multiple shooting method for real-time optimization of nonlinear DAE processes”. In: *Nonlinear model predictive control*. Springer. 2000, pp. 245–267.

[46] Weiwei Li and Emanuel Todorov. “Iterative linear quadratic regulator design for nonlinear biological movement systems”. In: *First International Conference on Informatics in Control, Automation and Robotics*. Vol. 2. SciTePress. 2004, pp. 222–229.

[47] Quoc Tran Dinh and Moritz Diehl. “Local convergence of sequential convex programming for nonconvex optimization”. In: *Recent Advances in Optimization and its Applications in Engineering: The 14th Belgian-French-German Conference on Optimization*. Springer. 2010, pp. 93–102.

[48] Riccardo Bonalli et al. “Gusto: Guaranteed sequential trajectory optimization via sequential convex programming”. In: *2019 International conference on robotics and automation (ICRA)*. IEEE. 2019, pp. 6741–6747.

[49] Tobia Marcucci et al. “Motion planning around obstacles with convex optimization”. In: *Science Robotics* 8.84 (2023), eadf7843.

[50] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *Int. J. Robotics Res.* 30.7 (2011), pp. 846–894. DOI: 10.1177/0278364911406761.

[51] Yanbo Li, Zakary Littlefield, and Kostas E. Bekris. “Asymptotically optimal sampling-based kinodynamic planning”. In: *The International Journal of Robotics Research* 35.5 (2016), pp. 528–564. DOI: 10.1177/0278364915614386.

[52] Ernesto Poccia. “Deterministic sampling-based algorithms for motion planning under differential constraints”. PhD thesis. Master’s thesis, Pisa Univ., Pisa, Italy, 2017.

[53] Wolfgang Hönig, Joaquim Ortiz de Haro, and Marc Toussaint. "db-A*: Discontinuity-bounded Search for Kinodynamic Mobile Robot Motion Planning". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*. IEEE, 2022, pp. 13540–13547. DOI: 10.1109/IROS47612.2022.9981577.

[54] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. "Real-time motion planning for agile autonomous vehicles". In: *Journal of guidance, control, and dynamics* 25.1 (2002), pp. 116–129.

[55] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. "Maneuver-based motion planning for nonlinear systems with symmetries". In: *IEEE Transactions on Robotics* 21.6 (2005), pp. 1077–1091.

[56] Matteo Saveriano et al. "Dynamic movement primitives in robotics: A tutorial survey". In: *The International Journal of Robotics Research* (2021).

[57] Edward Schmerling, Lucas Janson, and Marco Pavone. "Optimal sampling-based motion planning under differential constraints: the drift case with linear affine dynamics". In: *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE. 2015, pp. 2574–2581.

[58] Russ Tedrake. "LQR-Trees: Feedback motion planning on sparse randomized trees". In: *Robotics: Science and Systems* (2009).

[59] Anirudha Majumdar and Russ Tedrake. "Funnel libraries for real-time robust feedback motion planning". In: *The International Journal of Robotics Research* 36.8 (2017), pp. 947–982.

[60] Sertac Karaman and Emilio Frazzoli. "Optimal kinodynamic motion planning using incremental sampling-based methods". In: *49th IEEE conference on decision and control (CDC)*. IEEE. 2010, pp. 7681–7687.

[61] Wilko Schwarting et al. "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model". In: *IEEE Transactions on Intelligent Transportation Systems* 19.9 (2017), pp. 2994–3008.

[62] David Silver and Joel Veness. "Monte-Carlo Planning in Large POMDPs". In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty et al. Vol. 23. 2010.

[63] Zachary Sunberg and Mykel Kochenderfer. "Online algorithms for POMDPs with continuous state, action, and observation spaces". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 28. 2018, pp. 259–263.

[64] James Ragan, Benjamin Riviere, and Soon-Jo Chung. "Bayesian Active Sensing for Fault Estimation with Belief Space Tree Search". In: *AIAA Scitech*. Jan. 2023. DOI: 10.2514/6.2023-0874.

[65] Viliam Lisy et al. “Convergence of Monte Carlo Tree Search in Simultaneous Move Games”. In: *Advances in Neural Information Processing Systems*. Vol. 26. 2013.

[66] Marin Kobilarov. “Cross-entropy motion planning”. In: *The International Journal of Robotics Research* 31.7 (2012), pp. 855–871.

[67] Grady Williams et al. “Aggressive driving with model predictive path integral control”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1433–1440.

[68] Mohak Bhardwaj et al. “Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 750–759.

[69] Corrado Pezzato et al. “Sampling-based model predictive control leveraging parallelizable physics simulations”. In: *arXiv preprint arXiv:2307.09105* (2023).

[70] Neha Priyadarshini Garg, David Hsu, and Wee Sun Lee. “Despot-alpha: Online pomdp planning with large state and observation spaces.” In: *Robotics: Science and Systems*. Vol. 3. 2019, pp. 3–2.

[71] Yashwanth Kumar Nakka and Soon-Jo Chung. “Trajectory Optimization of Chance-Constrained Nonlinear Stochastic Systems for Motion Planning Under Uncertainty”. In: *IEEE Transactions on Robotics* (2022).

[72] Richard S. Sutton. “Planning by Incremental Dynamic Programming”. In: *Machine Learning Proceedings*. San Francisco, CA, 1991, pp. 353–357. DOI: https://doi.org/10.1016/B978-1-55860-200-7.50073-8.

[73] Thomas M Moerland et al. “Model-based reinforcement learning: A survey”. In: *Foundations and Trends in Machine Learning* 16.1 (2023), pp. 1–118.

[74] Chee-S Chow and John N Tsitsiklis. “An optimal one-way multigrid algorithm for discrete-time stochastic control”. In: *IEEE Transactions on Automatic Control* 36.8 (1991), pp. 898–914.

[75] Remi Munos and Andrew Moore. “Variable resolution discretization in optimal control”. In: *Machine learning* 49 (2002), pp. 291–323.

[76] Alex Gorodetsky, Sertac Karaman, and Youssef Marzouk. “High-dimensional stochastic optimal control using continuous tensor decompositions”. In: *The International Journal of Robotics Research* 37.2-3 (2018), pp. 340–377.

[77] Athanasios S Polydoros and Lazaros Nalpantidis. “Survey of model-based reinforcement learning: Applications on robotics”. In: *Journal of Intelligent & Robotic Systems* 86.2 (2017), pp. 153–173.

[78] Kai Arulkumaran et al. “Deep reinforcement learning: A brief survey”. In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[79] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[80] Maryam Fazel et al. “Global convergence of policy gradient methods for the linear quadratic regulator”. In: *International conference on machine learning*. PMLR. 2018, pp. 1467–1476.

[81] Kaiqing Zhang et al. “Global convergence of policy gradient methods to (almost) locally optimal policies”. In: *SIAM Journal on Control and Optimization* 58.6 (2020), pp. 3586–3612.

[82] Jalaj Bhandari and Daniel Russo. “Global optimality guarantees for policy gradient methods”. In: *Operations Research* (2024).

[83] Edward Schmerling et al. “Multimodal probabilistic model-based planning for human-robot interaction”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 3399–3406.

[84] Beomjoon Kim et al. “Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 9916–9924.

[85] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. “Model predictive path integral control: From theory to parallel computation”. In: *Journal of Guidance, Control, and Dynamics* 40.2 (2017), pp. 344–357.

[86] Richard S. Sutton, Doina Precup, and Satinder Singh. “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning”. In: *Artificial Intelligence* 112.1 (1999), pp. 181–211. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(99)00052-1.

[87] Aijun Bai, Siddharth Srivastava, and Stuart Russell. “Markovian State and Action Abstractions for MDPs via Hierarchical MCTS”. In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, New York, NY*. Ed. by Subbarao Kambhampati. IJCAI/AAAI Press, 2016, pp. 3029–3039. URL: http://www.ijcai.org/Abstract/16/430.

[88] Yiyuan Lee, Panpan Cai, and David Hsu. “MAGIC: Learning Macro-Actions for Online POMDP Planning ”. In: *Proceedings of Robotics: Science and Systems*. Virtual, July 2021. DOI: 10.15607/RSS.2021.XVII.041.

[89] Maarten De Waard, Diederik M Roijers, and Sander CJ Bakkes. “Monte carlo tree search with options for general video game playing”. In: *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE. 2016, pp. 1–8.

[90] Arec Jamgochian et al. *Constrained Hierarchical Monte Carlo Belief-State Planning*. 2023. arXiv: 2310.20054 [cs.AI].

[91] Richard S Sutton et al. “Policy gradient methods for reinforcement learning with function approximation”. In: *Advances in neural information processing systems* 12 (1999).

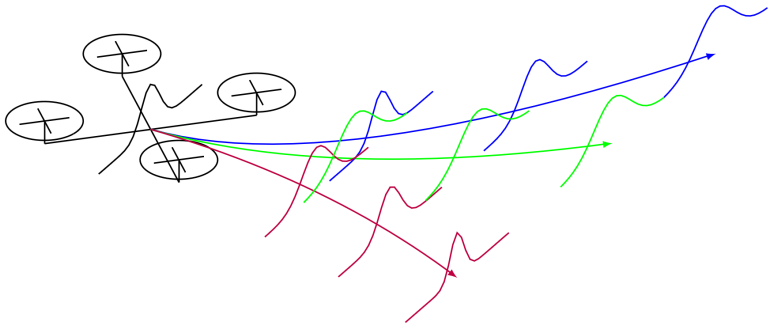
[92] Marc Deisenroth and Carl E Rasmussen. “PILCO: A model-based and data-efficient approach to policy search”. In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*. 2011, pp. 465–472.

[93] Sergey Levine and Vladlen Koltun. “Guided policy search”. In: *International conference on machine learning*. PMLR. 2013, pp. 1–9.

[94] Benjamin Rivière et al. “GLAS: Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning With End-to-End Learning”. In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4249–4256. DOI: 10.1109/LRA.2020.2994035.

Chapter 3

MCTS FOR BELIEF-SPACE PLANNING



How to construct trees and search over probability distributions?

This chapter is based on the publications:

James Ragan*, Benjamin Rivière*, and Soon-Jo Chung. "Bayesian Active Sensing for Fault Estimation with Belief Space Tree Search". In: *AIAA SciTech* (2023). DOI: 10.2514/6.2023-0874.

James Ragan, Benjamin Rivière, and Soon-Jo Chung. "Dreaming to Disambiguate: Safe Fault Estimation via Active Sensing Tree Search". In: *(Review at Science Robotics)* (2024).

The * denotes equal contribution.

3.1 Motivation

Autonomous robots operating independently of human-in-the-loop control offer the potential for dramatically increased capability by enabling faster operations and better performance in domains ranging from search and rescue [3] to planetary exploration [4]. However, a fully autonomous robot must be able to independently diagnose and recover from various component faults at a system level without waiting for outside guidance, especially when the robot's safety is time dependent, such as during autonomous driving [5] or when the system has a limited lifetime due to environmental degradation [6].

Spacecraft are motivating class of systems because time-delay and real-time au-

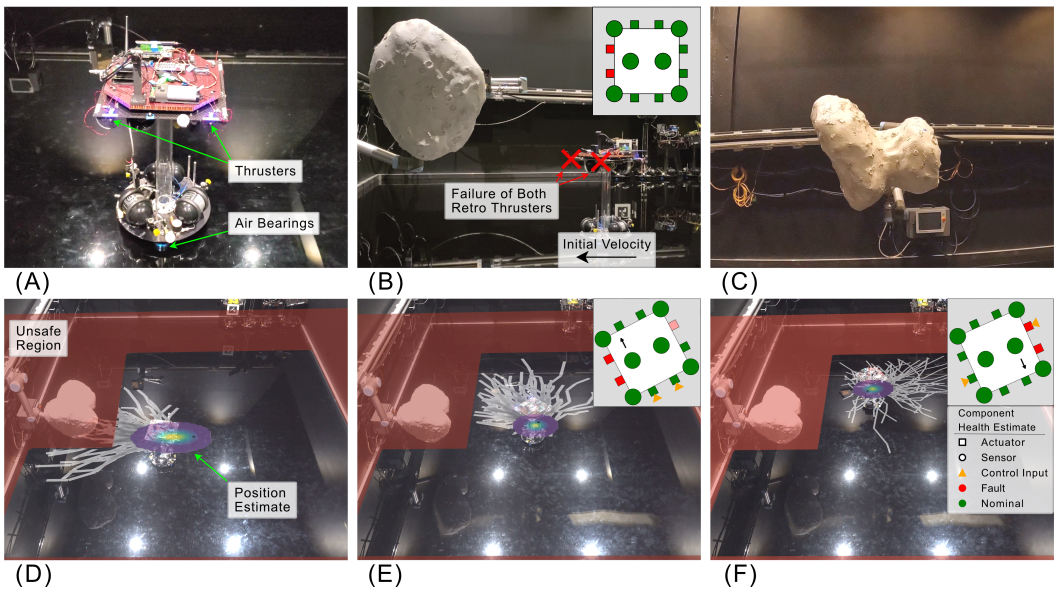


Figure 3.1: Safe fault estimation on robotic spacecraft.

tonomous operation makes human-in-the-loop interventions difficult if not impossible. As the number of space systems increase, so too do the number of failures. Over 2009-2016, the world launched an average of 46 small satellites per year, with 42.6% of launches terminating in failure or partial failure, with at least 27 of these lost missions attributed to actuator or sensor failure [7]. On Earth, unmanned aerial vehicles have a failure rate on the order of 1 in every 1000 hours of operation [8], though failures can occur much sooner in some domains and partial failures are reported to regularly occur as often; as every 10-50 hours [9].

To make robotic systems robust to these failures, previous work [10, 11] have used Fault Detection, Isolation and Recovery techniques (FDIR) to estimate the likelihood of failures from discrepancies in the system's onboard sensing. Work has been done to design ways to excite the system for more rapid fault diagnosis [12], although these methods are often limited to specific types of systems or disturbances, have limited or no ability to simultaneously consider safety, and can be prohibitively computationally expensive to react to real-time information. To ensure safety of a robot subject to state constraints, deterministic approaches often seek formal guarantees of safety [13] whereas probabilistic approaches typically try to satisfy a chance constraint over a horizon and solve approximately [14]. In either safety framework, the system model is usually assumed to be known, so no simultaneous fault estimation is considered.

We consider robotic spacecraft operating in time sensitive, safety critical environ-

ments, such as the approach to a small solar system body shown in Fig. 3.1, where component failure could jeopardize mission success. In this scenario, we envision a system level emergency response, where safely and autonomously identifying the underlying fault as quickly as possible supersedes previous mission objectives. To this end, we propose s-FEAST (Safe Fault Estimation via Active Sensing Tree search), a planning-based approach that selects diagnostic actions expected to excite informative observations for estimating the underlying fault while maintaining system safety, by considering probabilistic state constraints at each planning step. In Fig. 3.1, an autonomous robotic spacecraft undergoes a failure of both its retro thrusters while approaching a comet. Traditional passive fault detection approaches based on a mismatch between the actual and expected sensor observations or telemetry signals will likely not detect this failure until the spacecraft attempts to slow down, at which point it will be too late. 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 worst-case fault.

3.2 Problem Statement

First, we formalize the general partially observable optimal control problem and specify to our safe active fault estimation setting. Then, we present our algorithm including the marginalization filter, safety condition, and integration into tree search. Then, we present theoretical analysis of s-FEAST's optimality convergence. In the final subsection, we isolate the difference between tree-search with marginalization filtering (s-FEAST) and particle filtering (POMCP) in the context of information gathering problems.

Partially observable optimal control problem

The general partially observable optimal control problem provides the theoretical framework to study the active fault estimation problem. We consider control-affine system dynamics with linear sensing and additive noise processes are defined as:

$$x_k = f(x_{k-1}) + B(x_{k-1})u_k + w_k \tag{3.1}$$

$$y_k = Cx_k + v_k \tag{3.2}$$

where k subscript denotes a time index, $x \in X \subseteq \mathbb{R}^n$ is the physical state, $u \in U \subseteq \mathbb{R}^m$ is the control input, $f(x_k)$ is the unforced dynamics, $B(x_k)$ is the input influence matrix, $y \in Y \subseteq \mathbb{R}^p$ denotes the measurement, C is the measurement matrix, and the random process and measurement noise sequences w_k, v_k are assumed to be mutually independent and i.i.d. For simplicity, we will assume the noise processes are

Gaussian with covariance matrices Σ_w, Σ_v respectively, but in developing our safety condition in Sec. 4.4 and our algorithm in Sec. 4.5 we will show our algorithm is not restricted to only Gaussian noise.

In the presence of state uncertainty, it is common to write the probability distribution of the state as a belief, which can be computed by updating a prior with an observation and control input using Bayesian filtering [15]:

$$b_k(x) = \mathbb{P}(x_k \mid \overline{y}_k, \overline{u}_k) = \frac{\mathbb{P}(y_k \mid x_k) \int \mathbb{P}(x_k \mid x_{k-1}, u_k) b_{k-1}(x) \mathrm{d}x_{k-1}}{\mathbb{P}(y_k \mid \overline{y}_{k-1}, \overline{u}_k)} \tag{3.3}$$

where $b_0(x) = \mathbb{P}(x_0)$ is the prior and the overbar notation defines a history, e.g. $\overline{u}_K = \{u_1, \ldots, u_K\}$. The space of all possible beliefs is denoted $\mathcal{B}$.

The notion of optimality is defined through the reward function, the policy function, and the value function. We consider reward functions that are a map from belief to the scalar reals that specify the objective, and we assume the rewards are bounded between 0 and 1, $R : \mathcal{B} \to [0, 1]$. The policy function is a stochastic map from belief to action, $\pi : \mathcal{B} \to U$ and the set of all policies is denoted Π. For a finite horizon problem, the value function is the expected return of a policy from an initial belief:

$$V^{\pi}(b_0) = \mathbb{E}\left[\sum_{k=1}^{K} R(b_k) \mid \pi, b_0\right], \text{ s.t. Eqs. (3.1), (3.2), (3.3) } \forall k \tag{3.4}$$

where the kth control is generated from the policy, $u_k = \pi(b_{k-1})$, R is the problem's reward function, K is the horizon length and the expectation is over the stochastic policy, process and measurement noise processes.

Now, we define the partially observable optimal control problem [16]:

Definition 4 (Partially Observable Optimal Control) *The partially observable optimal control problem for the system given by Eqs.* (3.1)*,* (3.2)*, is to find the policy that maximizes the expected reward over the planning horizon from an initial belief* b_0*:*

$$V^*(b_0) = \max_{\pi \in \Pi} \; V^{\pi}(b_0) \tag{3.5}$$

Control policies are closed-loop solutions, selecting a new action at each time step in response to the belief updated via the new observation.

Safe active fault estimation problem formulation

Next we specify the general notation into our active sensing problem: to plan actions such that the resulting observations converge the belief of the underlying failure to the true failure as quickly as possible while maintaining safety. We define the following state and measurement equations, while restricting the control input to be in a discrete set U:

$$x_k = f(x_{k-1}) + B(x_{k-1})\left((\mathbb{I} - \Phi_B)u_k + \Phi_{B,\mathbb{1}}\right) + w_k \tag{3.6}$$

$$y_k = (\mathbb{I} - \Phi_C)Cx_k + \Phi_{C,\mathbb{1}} + v_k \tag{3.7}$$

$$u_k \in U \subseteq \{0,1\}^m \tag{3.8}$$

where, $\{0,1\}^m$ is the set of binary m-dimensional vectors. The system description differs from the control-affine system only in the fault model, Φ_B, Φ_C, $\Phi_{B,\mathbb{1}}$, $\Phi_{C,\mathbb{1}} = \text{diag}(\phi_{B/C/B,\mathbb{1}/C,\mathbb{1}})$ representing changes to actuator dynamics or sensing due to degradation or bias attacks in the actuators and sensors:

$$\phi_{B_i} = \begin{cases} 1 & \text{if } i \text{ actuator is completely failed} \\ 0 & \text{if } i \text{ actuator is nominal} \\ a_i & \text{if } i \text{ actuator is partially degraded} \end{cases}, \qquad \phi_B = [\phi_{B_1}, \ldots, \phi_{B_m}], \tag{3.9}$$

$$\phi_{B,\mathbb{1},i} = \begin{cases} 1 & \text{if } i \text{ actuator is stuck full on} \\ 0 & \text{if } i \text{ actuator is nominal} \\ a_i & \text{if } i \text{ actuator is partially biased} \end{cases}, \qquad \phi_{B,\mathbb{1}} = [\phi_{B,\mathbb{1},1}, \ldots, \phi_{B,\mathbb{1},m}] \tag{3.10}$$

where $a_i \in \{\delta_\phi, 2\delta_\phi, \ldots, 1 - \delta_\phi\}$ and $\delta_\phi \in (0,1)$ is the resolution of degradations considered. 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 paper. We assume the fault state does not change with time, so we drop the time subscript k from ϕ terms, and we define the concatenated vector of all faults:

$$\phi_k = \phi_{k-1} = \phi = (\phi_B, \phi_{B,\mathbb{1}}, \phi_C, \phi_{C,\mathbb{1}}) \in \Phi \subset [0,1]^{2(m+p)} \tag{3.11}$$

where Φ is the finite set of all possible faults that lives in the continuous space of $2(m+p)$ dimensional vectors with elements restricted between 0 and 1. We define the augmented state by composing the physical state and fault state, $q = [x; \phi]$

where $q \in Q = X \times \Phi$. Similar to (3.3), the belief and its update are defined as:

$$b_0(q) = \mathbb{P}(q_0), \quad b_k(q) = \mathbb{P}(q_k \mid \overline{y}_k, \overline{u}_k) = \frac{\mathbb{P}(y_k \mid q_k) \int \mathbb{P}(q_k \mid q_{k-1}, u_k) b_{k-1}(q) \mathrm{d}q_{k-1}}{\mathbb{P}(y_k \mid \overline{y}_{k-1}, \overline{u}_k)} \tag{3.12}$$

Next, we define the information gathering reward. First, we define the marginal beliefs over the failure space and physical states:

$$b_k(\phi) = \int_{x \in X} b_k(x, \phi) \mathrm{d}x, \quad b_k(x) = \sum_{\phi \in \Phi} b_k(x, \phi) \tag{3.13}$$

The information gathering reward and value function are then:

$$R(b_k, u_k) = \sum_{\phi \in \Phi} (b_k(\phi))^2 \tag{3.14}$$

$$V^{\pi}(b_0) = \mathbb{E}\left[\sum_{k=1}^{K} R(b_k) \mid \pi, b_0 \right], \text{ s.t. Eqs. (3.6), (3.7), (3.12)}, u_k = \pi(b_{k-1}) \; \forall k \tag{3.15}$$

This reward $R(b_k)$ corresponds to how confident the current belief is in the underlying fault state. 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. This reward function has previously been proposed as an uncertainty measure [17].

We use a standard superlevel set notion of probabilistic safety:

Definition 5 (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 $h(x) = \min_i g_i(x)$, 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 α :*

$$\mathcal{B}_{h,\alpha} = \{b \in \mathcal{B} \mid \int_X b(x) \mathbb{1}_{X_h}(x) \mathrm{d}x \geq \alpha\} \tag{3.16}$$

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

We can now define the Safe Active Sensing Fault Estimation problem:

Definition 6 (Safe Active Fault Estimation) *The safe active fault estimation problem for a given safety function, h and safety threshold α, modifies the original partially observable optimal control problem given by Definition 4, via Eqs.* (3.6)-(3.15) *with the additional constraint that each belief is α-safe.*

$$V^*(b_0) = \arg\max_{\pi \in \Pi} \; V^{\pi}(b_0) \quad \text{s.t.} \quad \mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi, b_0] = 1, \; \forall k \tag{3.17}$$

where the expectation is across the stochastic policy, measurement and process noise sequences.

3.3 Safe Fault Estimation with Active Sensing Tree Search

We present the s-FEAST algorithm and discuss the changes with respect to existing belief-space tree-search. We visualize our algorithm in Fig. 3.2A, and include the pseudocode in Algorithm 2.

In Fig. 3.2A, we show the diagram of the tree search applied to belief-space planning problems employed by s-FEAST. 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 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 $a_{1,1}$. In Fig. 3.2B we illustrate our marginalized filter representing the position of the robotic spacecraft conditioned on each possible failure. In Fig. 3.2C, we show the marginalized filter of a complicated multi-modal distribution can also be represented as a combination of Gaussians. In Fig. 3.2D, 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 a finite sample Chebyshev bound on 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.

Our algorithm s-FEAST is similar to an existing belief-space Monte Carlo Tree Search algorithm known as Partially Observable Monte Carlo Planning (POMCP) [18]. However, our approach is different because POMCP uses state samples to simultaneously estimate the belief and the optimal policy, whereas s-FEAST uses our

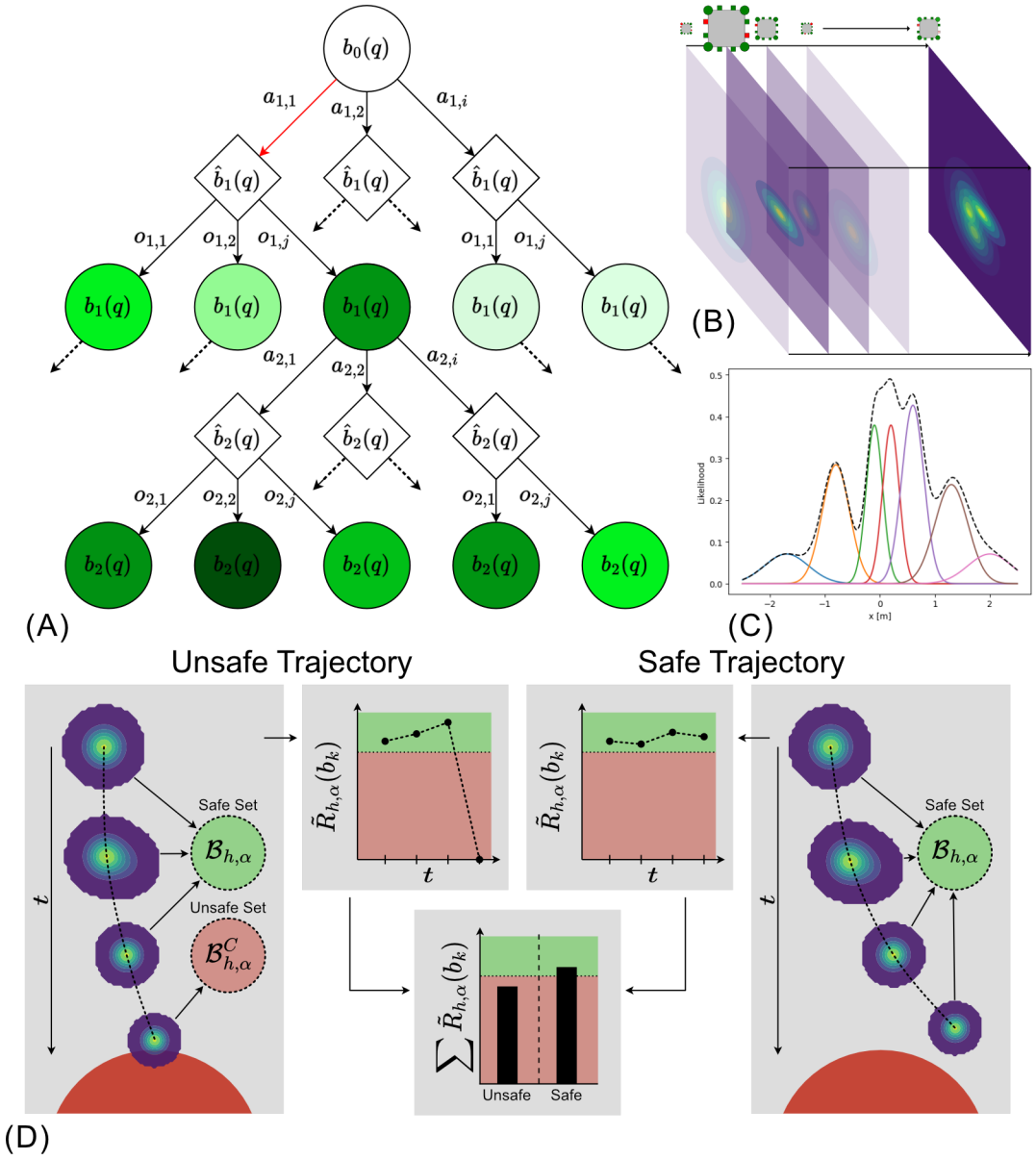


Figure 3.2: s-FEAST Method overview.

marginalization filter to immediately estimate the correct belief. Moreover, s-FEAST has a special treatment of the chance-constrained safety condition.

In our previous work [19] we empirically showed this 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 breaks down of POMCP breaks down. In this work, we formalize this observation.

POMCP consists of two components, Partially Observable UCT (PO-UCT), which assumes access to the state belief for a given history, and 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 UCT. At each node encountered during the simulation, the propagated particle is added to particle belief (or initializes a new one). The resulting belief at each node is represented as:

$$b_{\text{node}}(x) = \frac{1}{N_{\text{node}}} \sum_{i=1}^{N_{\text{node}}} \delta(x = x_i) \tag{3.18}$$

where x_i denotes the ith particle at the node for $i = 1, \ldots, N_{\text{node}}$ and δ is the Kronecker delta function. POMCP argues that, at large number of samples N, the belief is well approximated such that the UCT is solving the equivalent Belief Markov Decision Process (BMDP) and inherits the same value convergence of UCT. However Lemma 1 of [18], which equates the expected rewards for a POMDP and those of the corresponding derived BMDP, only establishes this for the PO-UCT algorithm, as this lemma assumes accurate state beliefs for a each history and rewards for each node. Neither is initially true in the information gathering setting, leading 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 of Eq. (3.14) is inversely correlated to the number of visits to the node, resulting in a breadth first search and random action selection. Further, standard particle filters are known to scale exponentially with the number of dimensions [20], exasperating this behavior.

In Algorithm 2, we show the pseudocode of the tree search methods. In black color, we show the original POMCP method and we highlight the changes for FEAST and s-FEAST in yellow and blue: (i) after a node is expanded, the exact Bayesian update is computed with our Marginalized filter, Eq. (3.19); and (ii) when rolling out

after encountering a new node, we compute the full Bayesian update to generate a value estimate; (iii) we approximate the safety at each node via our safety condition (Theorem 3). In the next two sections, we specify the marginalized filter and the safety condition.

Algorithm 2: 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 [18]. MF refers to our marginalized filter, approxSafety refers to the approximate safety condition given by Eq. (3.20) 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** $\arg\max_u \hat{V}(\{u\})$;

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** approxSafety($\hat{\mu}_h, \hat{\sigma}_h, M, \alpha$) ;

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**
  - $b(H_{d+1}) \leftarrow \mathrm{MF}(b(H_d), u_{d+1}, y_{d+1})$;
- **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**
  - $r \leftarrow \mathrm{safe}(b(H_{d+1}), h, \alpha)\,(r_0 + (1 - r_0)r)$;
- $r \leftarrow r + \gamma\ \mathrm{simulate}(q_{d+1}, H_{d+1}, d+1, b(H_{d+1}))$;
- **if** $N(H \cup u_{d+1}) = 0$ *AND* ***not*** *s-FEAST* **then**
  - return r ;
- $N(H) \leftarrow N(H) + 1$;
- $N(H \cup u_{d+1}) \leftarrow N(H \cup u_{d+1}) + 1$;
- $\hat{V}(H \cup u_{d+1}) \leftarrow V(H \cup u) + \frac{r - \hat{V}(H \cup u_{d+1})}{N(H \cup u_{d+1})}$;
- return r ;

Marginalized filter

We present our marginalization filter that is used in the s-FEAST algorithm to estimate the belief accurately. The key observation is that the dynamics (3.6) and measurement (3.7) of the active sensing problem have structure we can exploit to efficiently compute the belief update. Whereas jointly computing the belief update for the physical and fault state is 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). Any other nonlinear filtering approach can be used in lieu of EKF. The marginalization approach is similar to the Rao-Blackwellized filter used in FastSLAM [21], where the posterior is factored into estimations of each landmark that are conditioned on the robot path.

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

$$b_k(q) = \mathbb{P}([x_k, \phi] \mid \overline{y}_k, \overline{u}_k) = (\text{E})\text{KF}_\phi[y_k, u_k, b_{k-1}(x)](x_k)\frac{\tilde{Z}_\phi(y_k, u_k, b_{k-1}(q))b_{k-1}(\phi)}{\tilde{Z}(y_k, u_k, b_{k-1}(q))} \tag{3.19}$$

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, and the second term is a unconditional Bayesian update on each possible failure scenario where $\tilde{Z}_\phi$ and $\tilde{Z}$ are conditional and unconditional measurement likelihood functions. We compute $\tilde{Z}_\phi$ as the measurement relative likelihood given by the prediction step of each conditional EKF (before measurement innovation). We then note that $\tilde{Z}$ is a normalization factor, so does not need to be computed explicitly. The resulting filter is visualized in Fig. 3.2B, and resembles using multiple Gaussians to represent a complicated distribution as in Fig. 3.2C.

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 significantly 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 [22]. 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 useful to scale s-FEAST to high-dimensional systems with isolated faults, as only a subset of the estimation needs to be repeated for each considered fault.

Safety condition

In order to assure safety, we have 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, we use the following conservative sampling-based condition using concentration inequalities that is computationally efficient:

$$\frac{1}{M+1}\left\lfloor \frac{M+1}{M}\left(\frac{\hat{\sigma}_h^2(M-1)}{\hat{\mu}_h^2}+1\right)\right\rfloor \leq 1-\alpha \implies b \in \mathcal{B}_{h,\alpha} \tag{3.20}$$

where $\hat{\mu}_h$ and $\hat{\sigma}_h^2$ are the sample average and standard deviation resulting from sampling the safety condition $M > 2$ times for a given belief b. In the following subsection we derive this condition, and use it in our convergence analysis of s-FEAST.

3.4 Theoretical Result

The goal of our analysis is to show that the value estimate of Algorithm 2 converges to the solution of the safe active sensing for fault estimation problem, Def. 6. The logic is as follows: we reformulate the constrained problem into an equivalent unconstrained problem, and we transform the unconstrained problem again using the conservative safety condition, and finally we run s-FEAST on the resulting problem and inherit standard convergence guarantees. The proofs for these results are included in the Supplementary Materials. In the following, we consider constraints in the decision making problem sense [23]. In particular, by a constraint or safety constraint, we mean there are states or beliefs that are inadmissible. Conversely, an unconstrained problem has no inadmissible states.

First, we reformulate the constrained problem into an equivalent unconstrained problem. This step is necessary because standard Monte Carlo Tree Search (MCTS) techniques do not explicitly handle constraints. This argument is similar to that presented in convex optimization [24] 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:

$$R_{h,\alpha}(b_k) = \mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k)\,(r_0 + (1-r_0)R(b_k)) \tag{3.21}$$

$$V_{h,\alpha}^{\pi} = \mathbb{E}\left[\sum_{k=1}^{K} R_{h,\alpha}(b_k) \mid \pi, b_0\right] \text{ s.t. Eqs. (3.6), (3.7), (3.12)}\,, u_k = \pi(b_{k-1})\ \forall k \tag{3.22}$$

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

The first result is that the solution of the transformed problem is equivalent to the solution of the original problem (Definition 6), formalized with the following theorem:

Theorem 2 (Equivalent unbounded reformulation) *If a global optimal solution exists to the constrained safe active fault estimation problem, Definition 6, then the solution of the following unconstrained problem with a transformed value function given by Eq.* (3.22)*, is also a global optimal solution of Definition 6:*

$$\pi^*_{h,\alpha}(b_0) = \arg\max_{\pi\in\Pi} V^{\pi}_{h,\alpha}(b_0) \tag{3.23}$$

The proof is presented in the Supplementary Materials. Next, we develop our conservative approximation of $\mathbb{1}_{\mathcal{B}_{h,\alpha}}$. Our approach is based on the following finite sample approximation Chebyshev's Inequality, first developed in [25] and simplified in [26]:

$$\mathbb{P}\left(|Z - \hat{\mu}_Z| > \lambda\hat{\sigma}_Z\right) \leq \frac{1}{M+1}\left\lfloor \frac{M+1}{M}\left(\frac{(M-1)}{\lambda^2} + 1\right)\right\rfloor \tag{3.24}$$

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 [26], 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 ($\hat{\sigma}^2_h$) of this safety value, let $x_1, \ldots, x_M$ be i.i.d. samples of $b(x)$. We then have:

$$\hat{\mu}_h = \frac{1}{M}\sum_i h(x_i), \quad \hat{\sigma}^2_h = \frac{M+1}{M(M-1)}\sum_i (h(x_i) - \hat{\mu}_h)^2 \tag{3.25}$$

Our safety condition then follows directly from applying the finite sample Chebyshev inequality given by Eq. (3.24) 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, $\hat{\mu}_h, \hat{\sigma}_h$ defined according to Eq.* (3.25)*, and $\hat{\mu}_h \geq \hat{\sigma}_h$; satisfying the approximate safety condition of Eq.* (3.20) *(repeated below for reference) indicates that the belief is conservatively α-safe.*

$$\frac{1}{M+1}\left\lfloor \frac{M+1}{M}\left(\frac{\hat{\sigma}_h^2(M-1)}{\hat{\mu}_h^2}+1\right)\right\rfloor \leq 1-\alpha \implies b \in \mathcal{B}_{h,\alpha} \tag{3.20}$$

The proof is presented in the Supplementary Materials. 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. (3.20)). 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. (3.24) 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 3 (Conservative α-safe set) *For a 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:*

$$\frac{\sigma_h^2}{\mu_h^2} \leq 1-\alpha \iff b \in \tilde{\mathcal{B}}_{h,\alpha} \tag{3.26}$$

The proof is presented in the Supplementary Materials. To account for the slackness in our safety condition, we modify our reward function and present an additional *conservative* problem reformulation. We specify the reward and value functions:

$$\tilde{R}_{h,\alpha}(b_k) = \mathbb{1}_{\tilde{\mathcal{B}}_{h,\alpha}}(b_k)\,(r_0 + (1-r_0)R(b_k)) \tag{3.27}$$

$$\tilde{V}^{\pi}_{h,\alpha} = \mathbb{E}\left[\sum_{k=1}^{K} \tilde{R}_{h,\alpha}(b_k) \mid \pi, b_0\right] \text{ s.t. Eqs. (3.6), (3.7), (3.12)} \tag{3.28}$$

We present the final problem reformulation:

Definition 7 (Conservative Safe Active Fault Estimation) *The conservative safe active fault estimation problem is defined as follows:*

$$\tilde{\pi}^*_{h,\alpha}(b_0) = \arg\max_{\pi\in\Pi} \tilde{V}^{\pi}_{h,\alpha}(b_0) \tag{3.29}$$

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 6) exists and satisfies:*

$$\mathbb{E}[\mathbb{1}_{\tilde{\mathcal{B}}_{h,\alpha}}(b_k) \mid \pi, b_0] = 1 \ \forall k \tag{3.30}$$

where $\tilde{\mathcal{B}}_{h,\alpha}$ *is given by Lemma 3, then an optimal policy,* $\tilde{\pi}^*_{h,\alpha}(b_0)$*, to the conservative safe active fault estimation problem (Definition 7) is a sub-optimal solution of Definition 6 constrained to* $\tilde{\mathcal{B}}_{h,\alpha}$*.*

Further, if an optimal policy, $\pi^*(b_0)$*, to Definition 6 exists and satisfies Eq.* (3.30)*,* $\tilde{\pi}^*_{h,\alpha}(b_0)$ *is an optimal solution to Definition 6.*

The proof is presented in the Supplementary Materials. 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 6 and 7.

Theorem 5 (Optimality of s-FEAST) *Let* μ *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 7). In the limit of* $M \to \infty$*, the value of these policies converge:*

$$\lim_{N\to\infty} \left(\tilde{V}^{\mu}_{h,\alpha}(b_0) - \tilde{V}^*_{h,\alpha}(b_0)\right) \to 0 \tag{3.31}$$

with convergence rate $O(\log N/N)$*. Further, if an optimal policy,* $\pi^*(b_0)$*, to Definition 6 exists and satisfies Eq.* (3.30)*,* $V^{\mu}(b_0)$ *converges to* $V^*(b_0)$*.*

The proof is presented in the Supplementary Materials. In this section, we have reformulated the safe active fault estimation problem (Definition 6) to an unconstrained form by Theorem 2. We then used Theorem 3 and Lemma 3 to define a conservative sampling bound and the corresponding $\tilde{\mathcal{B}}_{h,\alpha}$ to define the conservative safe active fault estimation problem (Definition 6). Finally Theorem 4 formalizes when the solution to the two problems are equivalent, and Theorem 5 demonstrates convergence of s-FEAST to optimal solutions for each.

We make some remarks on this result: First, despite applying the existing search result from [27] and [18], 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 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 most risky trajectories. As adding safety at the cost of some performance is usually desirable in operation, and in regards to the previous observation that the approximation is not overly conservative, we believe this sub-optimal algorithm balances well the competing interests of safety, performance, and computational complexity.

3.5 Simulation Result

To validate our algorithm quantitatively, we consider 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 is performed on a 3 degree of freedom model of the M-STAR robot, and we evaluate each algorithm over 1000 trials according to the fraction of trials safe throughout the experiment and the product of a diagnostic reward and success rate. Details on these evaluation metrics, along with system descriptions and additional numerical results are provided in the Supplementary Materials.

Overview of baselines

We provide a brief overview of the baselines (random, greedy, D-CBF, SCP) we compare against in the upcoming simulations. The selection of baselines is designed such that s-FEAST and these baselines cover a permutation of deterministic vs. probabilistic state representations and greedy vs. planning algorithmic implementations, with s-FEAST as the probabilistic planning solution. All methods use the same estimator between time steps and each baseline solves for the next action to take. Implementation details are provided in the Supplementary Materials.

The first baseline is to randomly selecting actions uniformly from the admissible control set U. This method performs no optimization for information gathering or safety. The second baseline is a greedy, active approach with a probabilistic state representation. It simulates each action once, and selects a safe action with the best immediate reward computed with the marginalized filter, but only considers a lookahead horizon of one and does not resample any actions, making it vulnerable to near term danger and outlier simulations. Together, the random and greedy baselines serve 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 are deterministic safe control methods. The discrete control barrier function (D-CBF) [28] method acts greedily, considering safety of the next time step, whereas the sequential convex programming (SCP) [29, 30] 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. To adapt them to our partially observable setting, we use the most likely failure state and its mean position estimate as the assumed system dynamics and initial position and add 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 see this method fail in our simulations for this reason. Like D-CBF based methods, SCP needs a deterministic system dynamics model, but plans the next actions over a horizon. Work has been done to extend both methods to consider stochastic noise via chance constraints for SCP [31] and probabilistic safety bounds for D-CBF [32], though neither method is compatible with the coupled fault mode and physical state uncertainty considered here. These control baselines serve to illustrate the limitations of the control-estimation separa-

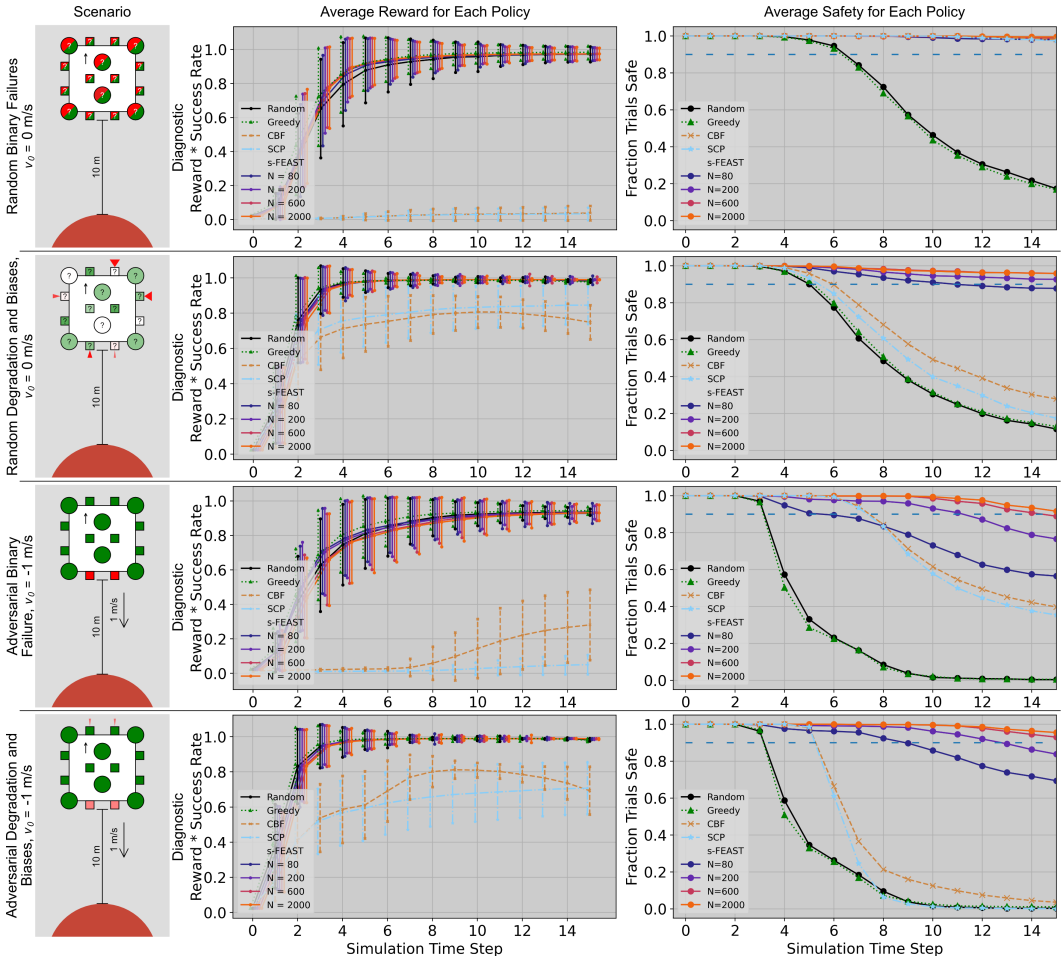


Figure 3.3: Validation of s-FEAST: The numerical performance of our algorithm compared with baselines across several scenarios.

tion principle in safety-critical fault estimation problems.

Overview of scenarios

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

To highlight various sources of difficulty our method addresses, we consider two

fault cases in two increasingly difficult initial conditions. Binary faults, where components either work or are completely failed, illustrate the challenges posed when components fail silently, resulting in ambiguity between fault models. Alternatively, continuous component degradation and biases present a larger challenge for safety, as actuator biases can destabilize a system if unaddressed.

Scenario: binary fault diagnosis in proximity to an obstacle

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

Examining the safety of each method, we see that the greedy and random baselines dramatically underperform 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. The greedy method's slightly worse safety is not surprising. On average, the greedy algorithm fired 3.73 thrusters for each action, whereas the random method fired 3.66. This 2.0% increase correlates with the 2.4% decrease in final safety values. A plausible explanation is that actions with larger control inputs were more likely to excite observations leading to a better belief update, and were therefore selected more often by the greedy method. But these also are more likely to leave the system in a higher velocity state, making collisions with an obstacle more likely. We see this trend continue in other experiments.

Considering the reward and diagnostic success of each method, the s-FEAST algorithms all outperform the CBF and SCP baselines, as do the random and greedy baselines. This is due to the CBF and SCP baselines not taking any information gathering actions, or any actions at all until the system is close to becoming unsafe. So both baselines typically fail to diagnose the underlying failure by the end of the experiment, leading to low diagnosis success rates of 20.8% and 19.5% respectively. The random and greedy algorithms perform similarly to the s-FEAST algorithms in diagnosing the underlying fault, but at the price of significantly worse safety, with final safety values of 17.4% and 17% respectively.

Scenario: continuous degradation & biases fault diagnosis in proximity to an obstacle

In this experiment, we consider the same scenario as before, but now components can 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 can 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 assume the fault is constant for the duration of our diagnosis period. Faults are generated by sampling 8 unique biases with 5 component degradations each, for a total of 40 possible faults as before. The true fault is set to one of these. Details of the fault model and experimental set ups are provided in Materials and Methods section and the Supplementary Materials respectively.

The results of this simulation are shown in the second row of Fig. 3.3. Compared to the previous scenario, we see similar relative behavior, and all methods have a higher diagnostic reward and a lower safety. This is because the faults are no longer silent. Any bias injects a signal into the system to enable passive identification of these faults. However, the active signal makes enforcing safety more challenging, as bias acceleration can lead to constraint violations. This trade off is seen through a drop in safety for all policies. For example, from time step 3 to 4, the deterministic methods (SCP and CBF) start to decline in safety, whereas the diagnostic reward increases more rapidly than the s-FEAST methods for the first time. This trend continues 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 cannot be explicitly performed. This can result in an incorrect assessment of both the safety of the current state as well as the control authority if actuator faults are not yet detected or resolved. When actions are taken to avoid collision, they may occur 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 is 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 failures

In the final two scenarios examined, the spacecraft now is subject to the same underlying fault in every trial and is initialized on a collision course with an obstacle. We consider an adversarial failure for both our binary and continuous degradation and bias scenarios. In the binary scenario, the two retro thrusters on the spacecraft are completely off, and in the continuous case, the retro thrusters are subject to an 80% degradation and the forward thrusters are subject to a 10% bias. In both cases, the spacecraft must first change orientation, then slow down to reliably avoid a collision. Because this behavior requires planning over a horizon, we consider these to be adversarial faults for this scenario and choose this scenario to demonstrate s-FEAST's robustness to outlier failures that pose significant risk to the system. The spacecraft still starts with a uniform prior over 40 possible failures so must also take actions to reduce the risk of collision while fully identifying the underlying fault.

The results are shown in the bottom two rows of Fig. 3.3, where we see that all baselines now achieve less than 40% safety in the binary case (CBF: 37.2% , SCP: 35.8%, Random: 1.8%, Greedy: 1.8%) and less than 4% in the continuous case (CBF: 3.6% , SCP: 0.1%, Random: 0.3%, Greedy: 1.1%), and are 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 59.1% in the binary case, 67.2% in the continuous case. For $N = 200$, the safety rate is 79.5% and 86.8% for the binary and continuous faults. We use this result to inform our real-time hardware experiments, where the typical planning amount is $N = 85$ simulations per tree due to a tight computational budget. The reward for the hardware algorithm shown in Fig. 3.1 is similar to that predicted by this simulation experiment.

We again see that with the binary faults, our CBF and SCP baselines fail to gather any information until collision is imminent, and only gain diagnostic reward as a result of attempting to remain safe. Similarly, in the continuous case, the baseline methods gain some information immediately as a result of the bias signal, but fail to further diagnose until evasive actions are taken, which occurs 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 succeeds where these baseline methods fail.

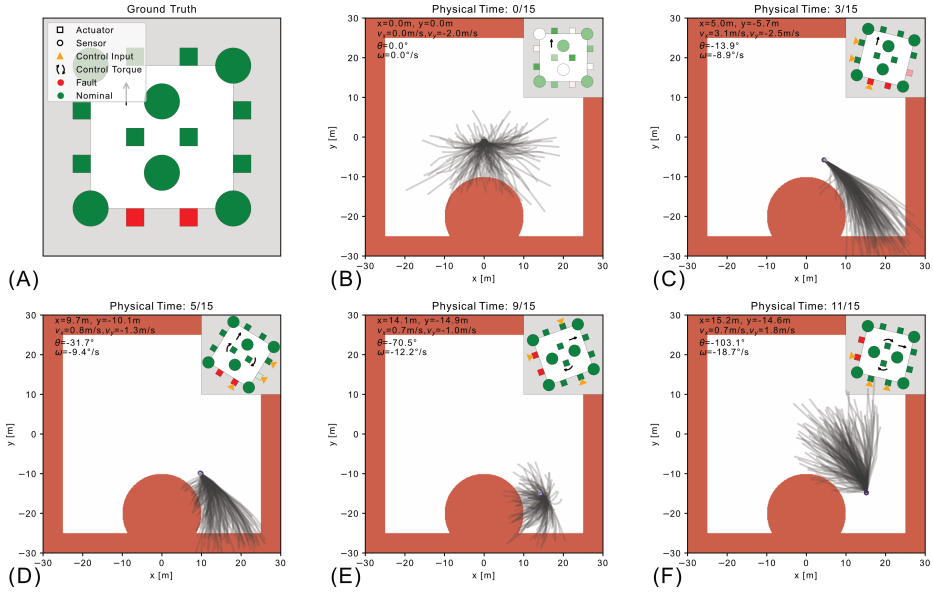


Figure 3.4: Qualitative analysis of s-FEAST's collision avoidance under an adversarial fault.

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 selects actions to adjust the spacecraft's trajectory to the side of the obstacle. This turns out to be a necessary strategy in this scenario, as after the failure is identified in the third time step, it takes another seven time steps to reorient and come to a stop. This obstacle avoidance behavior is also seen in our hardware experiments, such as in Fig. 3.1.

The baseline methods are 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.

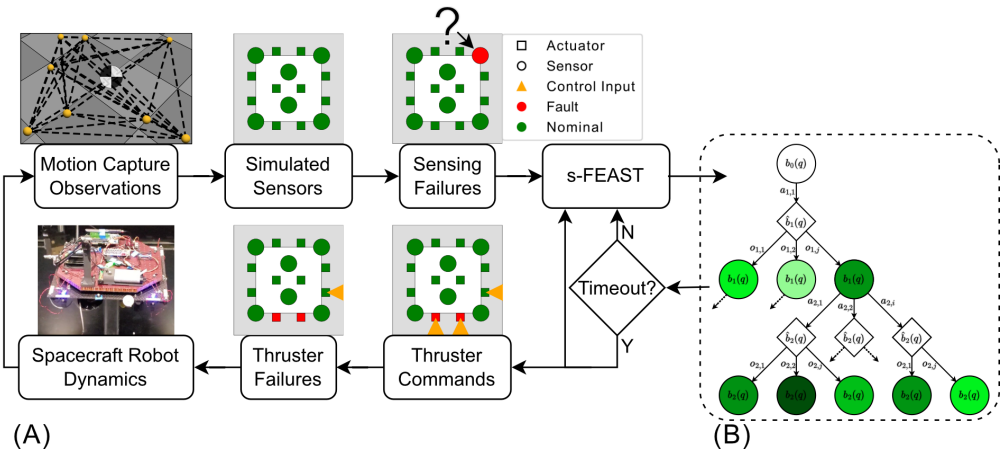


Figure 3.5: Real-time implementation of s-FEAST

3.6 Hardware Result

We implement s-FEAST on the M-STAR robot [33, 34] using the Caltech Autonomous Robotics and Control Lab's (ARCL) spacecraft simulator facility, shown in Fig. 3.1A. The M-STAR robot floats on air bearings on a high precision flat floor to create a very low friction environment and simulate spacecraft dynamics, and is actuated using thrusters. A motion capture system provides position and orientation measurements and noise is artificially added according to our observation model, shown in Fig. 3.5A.

The robot is tasked to diagnose sensing and actuation faults while on a collision course with our model comet (Fig. 3.1C). The true failure is the loss of both retro thrusters (Fig. 3.1B), requiring the M-STAR robot to reorient before it is able slow down and stabilize itself. The safety constraints are to avoid the comet obstacle as well as the walls of the simulator room, shown as the red regions in Fig. 3.1D-F, with a 90% or higher probability. With these settings, s-FEAST is able to successfully identify the true failure state while maintaining safety, validating our approach on hardware. The video of s-FEAST and baselines running on the M-STAR robot hardware is provided in Movie 1. Still frames are presented in Fig. 3.1, demonstrating that considering safety or fault estimation alone cannot solve this problem (Fig. 3.1D), and that s-FEAST can reliably plan evasive actions under uncertain component failure (Fig. 3.1E,F). A complete time series of s-FEAST and the baseline methods is presented in the Supplementary Materials.

To deploy s-FEAST in the real-time setting, we implement s-FEAST in a receding horizon fashion, i.e. the planner recomputes a policy every time step, and applies only the first action to the physical system. Because the dynamics continue

to propagate during the planning computation time, the state of the system when the planner began solving, x_k is different from the state when the selected action is taken, $x_{k+\delta_t}$ where δ_t is the propagation time. To synchronize these two states, we run the same s-FEAST algorithm, except we plan the next action to take from the expected result of the current action. The modified tree topology is visualized in Fig. 3.5B. Instead of specifying a number of simulations to run, we take advantage of s-FEAST's ability to provide an anytime solution by simulating until the computation budget is exhausted and returning the best action. The selected action is then applied onboard the robot and the current observation is used by s-FEAST to compute the next action to take while the system dynamics propagate. For these experiments, a budget of 0.78 seconds on a 1.10 GHz, 4 core CPU (i5-1035G4) was used, typically resulting in 85 simulations per time step. We show in numerical experiments this is sufficient computation to significantly improve over existing approaches.

3.7 Discussion

Comparison with POMCP

MCTS has been extended to solve POMDPs via the POMCP algorithm [18]. However, in our previous work [19] 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 POMCP breaks down. In this work, 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. The resulting belief at each node is a discrete collection of state particles, one for each visit to the node.

POMCP argues that at large number of samples N, the belief is well approximated such that the PO-UCT is solving the equivalent Belief Markov Decision Process (BMDP) and therefore inherits the value convergence of the fully observable UCT [27]. However, it only establishes this for the PO-UCT algorithm, as the theo-

retical analysis assumes accurate state beliefs for each history and accurate rewards for each node. Neither is initially true in the information gathering setting, leading 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 introduce in the next section (Eq. (3.14)) is inversely correlated to the number of visits to the node, resulting in a breadth first search where the MCTS 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 [20]. The difference in tree growth between s-FEAST and POMCP is visualized qualitatively in Fig. 3.6. In the Supplementary Materials, we provide additional numerical experiments to validate these claims. Pseudocode of the two algorithms is presented in the Materials and Methods.

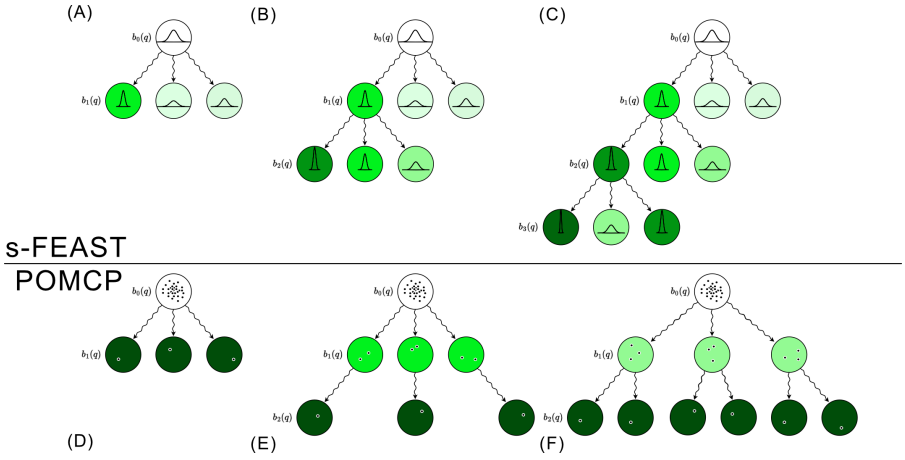


Figure 3.6: A conceptual comparison of the tree growth of s-FEAST and POMCP.

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 of the numerical results section. Instead, 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 faster run time and better performance.

To this end, we note s-FEAST presents two promising opportunities for future performance improvements. First, the marginalized filter we present in the Materials and Methods section 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 [35] and GPU use [36]. We view these types of optimizations as complementary to s-FEAST's algorithmic innovations to 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 bottom rows of Fig. 3.3.

The anytime property is also is 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 [37]. Instead, we can return the best solution found within any computation budget. We show in the next section that s-FEAST converges asymptotically to the optimal solution, a guarantee not provided by existing chance constrained anytime methods [38].

We expect the ongoing trend of ever increasing computational power onboard space robotics missions [39, 40] to be enabling for our methodology, especially as payloads are developed for increasingly data-intensive science applications. 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.

Application of s-FEAST to other information gathering problems

The active fault estimation approach we consider is most useful in systems with high functional redundancy that creates ambiguity between possible failures and also provides the ability to recover when the fault is identified. Our method will fail if a safety critical state of the system becomes completely unobservable or un-

controllable. However, these situations will be unrecoverable for our baselines and related work methods as well.

Finally, we note that our approach belief-space planning and sampling-based safety can be applied to other information-gathering problems where the underlying state has a computable belief-transition. For example, we can consider the classic problem of a robot autonomously mapping an unknown environment [41] or future robotic planetary exploration missions where actions are taken to scout out areas of potentially high scientific value [42]. In both cases, gathering information is a key goal, necessitating belief-space planning. And in both cases, the robot might be subject to additional constraints, ranging from the safety constraints we consider here, to battery or time budgets limiting exploration, where the effect of high variance makes constraints on expectation alone limiting.

3.8 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. 3.7. 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 be run in real-time. We elaborate on the relevant related work in the following subsections.

Traditional passive fault detection, isolation, and recover methods

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 [10, 11]. 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 [43]. Similarly for terrestrial robotics, fault estimation methods were historically passive and designed to alert operators and arrest operations [44], and have been limited by the high levels of ambiguity [45]. In more recent work, passive fault estimation in robotic systems has developed to include data-driven models and account for varying levels of

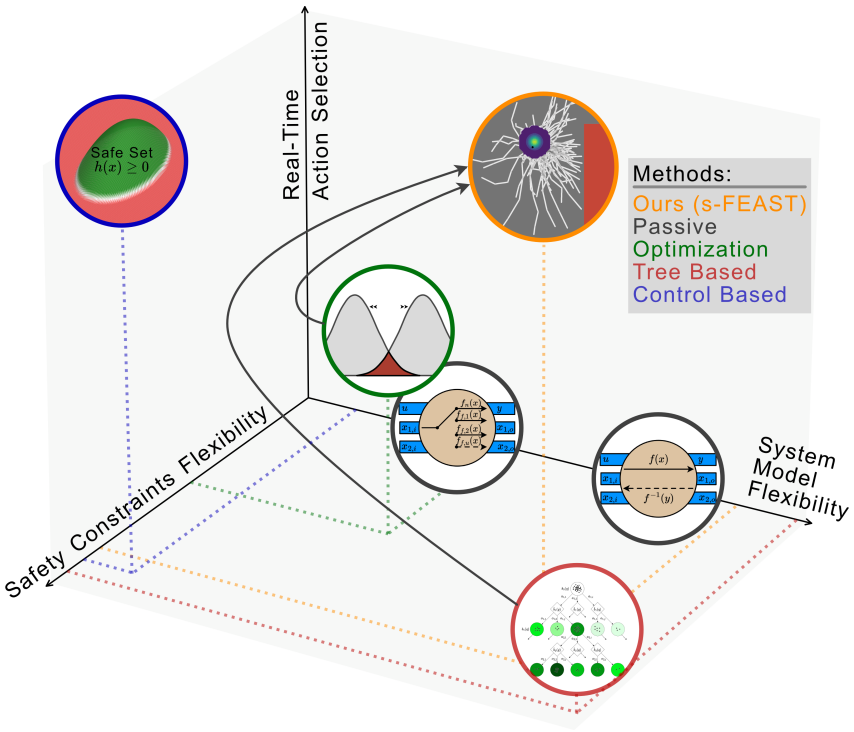


Figure 3.7: Related work in fault estimation.

system autonomy [46–48] and distributed systems [49]. 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. 3.7.

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 [50, 51]. While 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. Because 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: optimization methods

Passive fault estimation methods can return multiple plausible fault scenarios. A key idea is to resolve this ambiguity by taking actions to "induce" informative observations. Active Fault Diagnosis (AFD) approaches offer faster belief conver-

gence by selecting control inputs to excite useful observations [12]. Optimality conditions of active sensing can be derived [52], however general tractable algorithms do not exist. This has given rise to a large body of work on the best input designs for active fault diagnosis [12]. Early work considered discrete-event systems with enumerated states and transitions, designing controllers to ensure the system would be diagnosable [53], including for satellite systems [54, 55].

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 [56]. 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 [57] 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 [58]. 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 [37]. 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 [59], and this approach has been extended to include small, bounded non-linearities [60] and linearizations [61].

These AFD approaches demonstrate the usefulness of information gathering applied to fault estimation, and are represented as optimization methods in Fig. 3.7. 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. 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.

While not directly derived from this body of work, robotic self-modeling, where an robot continuously performs exploratory actions to update its onboard dynamics model [62] is a closely related method. Recently, self-identification has been used to distinguish between multiple possible manipulation models of a robotic hand [63], as well as to learn visual self models [64].

Partially observable Markov decision processes and tree-based methods

Partially Observable Markov Decision Processes (POMDPs), provide an alternative framework to consider from a system level both fault estimation and safety as decision-making problems. A POMDP is used to model the fault estimation problem in [65]. 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. This is a common trend in POMDPs, where solutions are often limited to small problems, or to being performed offline or inexactly [66].

To approximately solve decision-making problems online, we consider a family of tree-based planners known as Monte Carlo Tree Search (MCTS) [27]. MCTS approximates the solution to the fully observable Markov Decision Problem (MDP) problem by simulating future state trajectories while biasing the tree towards areas of high reward [67]. This has been extended to partially observable settings by Partially Observable Monte Carlo Planning (POMCP) [18], which uses simulated state trajectories to simultaneously estimate the optimal solution and the belief distribution. In the context of FDIR, decision trees have also been used to diagnose faults in systems by incorporating series of decisions, tests, or temporal information [68, 69].

To consider constraints in POMDPS, prior work has added cost terms that must be kept within a specified budget [23]. 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 [70], dynamic programming [71], and gradient ascent with constraint projection [72]. 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 [73], an extension of POMCP to discrete constrained POMDPs [14], and a method that extends online solutions to continuous systems by limiting branching [74]. A shared limitation is that these methods constrain only the expected

cost, which may not be suitable for risk adverse 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 [75], and allows for bounds on statistics other than expectations. Approximate offline solutions in this setting include using mixed integer linear programming [76], and pruning high risk branches [75]. To solve chance constrained POMDPs online, a heuristic search that is then iteratively improved in an anytime fashion has been proposed [38], but lacks formal guarantees. Chance constraints have also been applied in belief-space planning for linear Gaussian systems [77] and via log barrier function transformations or soft constraints [78].

POMDPs are also solvable with model-free approaches trained during an offline phase, such as Dreamer V2 [79] and latent policy optimization [80]. However, compared to online methods, the reliance on an offline training phase makes these methods vulnerable to out-of-domain events [81]. Furthermore, these methods lack theoretical guarantees of optimality convergence and safety assurance, which are especially important for high-cost space missions.

While tree-based online POMDP solvers such as POMCP [18] work well when the reward is a function of the state, experiments in our prior work [19] suggest they scale poorly 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. This results in the low real-time performance for this setting depicted in Fig. 3.7. By introducing a marginalized filter, our approach addresses this issue. The ability to recover efficient tree exploration is one of our key contributions, and we provide in-depth theoretical arguments for why this modification is necessary in the Discussion section as well as numerical validation in the Supplementary Materials. We also provide formal guarantees of constraint satisfaction and general bounds on tail probabilities as opposed to constraints on expectations alone. In this fashion, we consider our method to be a combination of the desirable theoretical properties and generality of POMDP methods with the model distinguishing capabilities of Active Fault Diagnosis.

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. Although these information gathering POMDPs have also been explored, previous work 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. 3.7. POMDP solvers can be guided toward information gathering behavior via sub-goal states, identified by heuristics measuring the entropy of measurement probabilities [82],[83]. 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 [84], 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 [85]. This approach has also been extended to continuous settings by augmenting the reward with convex information measures on the belief-space [86] which includes the information gathering reward we consider in our work.

Safety aware control methods

The final category of related work is optimal control-based approaches to safety. These include control barrier functions (CBFs), which formalize safety for deterministic and fully observable systems by providing necessary and sufficient conditions on a controller's ability to ensure safety for every admissible state [13]. These have recently been extended to discrete time systems [28], including those subject to stochastic noise [32]. In the face of randomly changing environmental hazards, metrics to quantify the risk an robot faces can be used to plan safe trajectories, such as using entropic value at risk to bound tail probabilities [87]. Other planning-based approaches to real-time optimal control with safety constraints include Sequential Convex Programming (SCP) [29, 30] which can consider complicated constraints over horizons [88] and has been extended to systems involving nonlinear stochastic dynamics and chance constraints [31]. Also, such planning-based approaches can be combined with tracking control for robustness and stability guarantees [89].

In partially observable settings, safety of a state cannot be directly observed. However similar probabilistic bounds on the belief of the system's state can be established. In [90], the system state is estimated with a particle filter, and the controller is designed to keep this estimate within a set of safe beliefs defined by 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. In [91], uncertainty arises from a number of possible system modes with differing dynamics, and actions are planned for that are safe for all possible modes using CBFs, for as long of a planning horizon as possible. 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. 3.7. 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.

BIBLIOGRAPHY

[1] James Ragan*, Benjamin Rivière*, and Soon-Jo Chung. "Bayesian Active Sensing for Fault Estimation with Belief Space Tree Search". In: *AIAA SciTech* (2023). DOI: 10.2514/6.2023-0874.

[2] James Ragan, Benjamin Rivière, and Soon-Jo Chung. "Dreaming to Disambiguate: Safe Fault Estimation via Active Sensing Tree Search". In: *(Review at Science Robotics)* (2024).

[3] David C Schedl, Indrajit Kurmi, and Oliver Bimber. "An autonomous drone for search and rescue in forests using airborne optical sectioning". In: *Science Robotics* 6.55 (2021), eabg1188.

[4] Vandi Verma et al. "Autonomous robotics is driving Perseverance rover's progress on Mars". In: *Science Robotics* 8.80 (2023), eadi3099.

[5] Tasuku Ishigooka, Shinya Honda, and Hiroaki 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.

[6] Alan Mantooth, Carl-Mikael Zetterling, and Ana Rusu. "Venus calling silicon carbide radio circuits can take the heat needed to phone home from our hellish sister planet". In: *IEEE Spectrum* 58.5 (2021), pp. 24–30.

[7] Stephen A Jacklin. *Small-satellite mission failure rates*. Tech. rep. NASA Ames Research Center, 2019.

[8] Federal Aviation Authority. *FAA Aerospace Forecast: Fiscal Years 2019-2039*. 2019.

[9] Matthew Osborne 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.

[10] Inseok Hwang et al. "A survey of fault detection, isolation, and reconfiguration methods". In: *IEEE Transactions on Control Systems Technology* 18.3 (2009), pp. 636–653.

[11] Alexandra Wander and Roger 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.

[12] Tor Aksel N. Heirung and Ali Mesbah. "Input design for active fault diagnosis". In: *Annual Reviews in Control* 47 (2019), pp. 35–50. ISSN: 1367-5788.

[13] Aaron D. Ames et al. “Control Barrier Functions: Theory and Applications”. In: *2019 18th European Control Conference (ECC)*. 2019, pp. 3420–3431.

[14] Jongmin Lee et al. “Monte-Carlo Tree Search for Constrained POMDPs”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[15] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005.

[16] J. Speyer, J. Deyst, and D. Jacobson. “Optimization of stochastic linear systems with additive measurement and process noise using exponential performance criteria”. In: *IEEE Transactions on Automatic Control* 19.4 (1974), pp. 358–366. DOI: 10.1109/TAC.1974.1100606.

[17] Georg Süssmann. “Uncertainty Relation: From Inequality to Equality”. In: *Zeitschrift für Naturforschung A* 52.1-2 (1997), pp. 49–52.

[18] David Silver and Joel Veness. “Monte-Carlo planning in large POMDPs”. In: *Advances in neural information processing systems* 23 (2010), pp. 2164–2172.

[19] James Ragan, Benjamin Riviere, and Soon-Jo Chung. “Bayesian Active Sensing for Fault Estimation with Belief Space Tree Search”. In: *AIAA Scitech 2023 Forum*. 2023, p. 0874.

[20] Simone Carlo Surace, Anna Kutschireiter, and Jean-Pascal Pfister. “How to Avoid the Curse of Dimensionality: Scalability of Particle Filters with and without Importance Weights”. In: *SIAM Review* 61.1 (2019), pp. 79–91.

[21] Michael Montemerlo et al. “FastSLAM: A factored solution to the simultaneous localization and mapping problem”. In: *Eighteenth National Conference on Artificial Intelligence* 593598 (2002).

[22] Joseph Marino, Milan Cvitkovic, and Yisong Yue. “A General Method for Amortizing Variational Filtering”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[23] Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

[24] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[25] John G Saw, Mark CK Yang, and Tse Chin Mo. “Chebyshev inequality with estimated mean and variance”. In: *The American Statistician* 38.2 (1984), pp. 130–132.

[26] Ata Kaban. “Non-parametric detection of meaningless distances in high dimensional data”. In: *Statistics and Computing* 22 (2012), pp. 375–385.

[27] Levente Kocsis and Csaba Szepesvári. "Bandit based monte-carlo planning". In: *European Conference on Machine Learning*. Springer. 2006, pp. 282–293.

[28] Ayush Agrawal and Koushil Sreenath. "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation." In: *Robotics: Science and Systems*. Vol. 13. Cambridge, MA, USA. 2017, pp. 1–10.

[29] Daniel Morgan et al. "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming". In: *The International Journal of Robotics Research* 35.10 (2016), pp. 1261–1285.

[30] Daniel Morgan, Soon-Jo Chung, and Fred Y. Hadaegh. "Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming". In: *Journal of Guidance, Control, and Dynamics* 37.6 (2014), pp. 1725–1740.

[31] Yashwanth Kumar Nakka and Soon-Jo Chung. "Trajectory Optimization of Chance-Constrained Nonlinear Stochastic Systems for Motion Planning Under Uncertainty". In: *IEEE Transactions on Robotics* (2022).

[32] Ryan K Cosner et al. "Robust Safety under Stochastic Uncertainty with Discrete-Time Control Barrier Functions". In: *arXiv preprint arXiv:2302.07469* (2023).

[33] Yashwanth Kumar Nakka et al. "A six degree-of-freedom spacecraft dynamics simulator for formation control research". In: *AAS/AIAA Astrodynamics Specialist Conference*. AIAA, 2018.

[34] Rebecca Foust et al. "Autonomous In-Orbit Satellite Assembly from a Modular Heterogeneous Swarm". In: *Acta Astronautica* 169 (Jan. 2020). DOI: 10.1016/j.actaastro.2020.01.006.

[35] Semanti Basu et al. "Parallelizing POMCP to solve complex POMDPs". In: *Robotics: Science and Systems (RSS) Workshop on Software Tools for Real-time Optimal Control*. 2021.

[36] Panpan Cai et al. "HyP-DESPOT: A hybrid parallel algorithm for online planning under uncertainty". In: *The International Journal of Robotics Research* 40.2-3 (2021), pp. 558–573.

[37] Davide M Raimondo et al. "Closed-loop input design for guaranteed fault diagnosis using set-valued observers". In: *Automatica* 74 (2016), pp. 107–117.

[38] Sungkweon Hong et al. "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.

[39] Bob Balaram et al. “Mars helicopter technology demonstrator”. In: *2018 AIAA Atmospheric Flight Mechanics Conference*. 2018, p. 0023.

[40] Windy S. Slater et al. “Total Ionizing Dose Radiation Testing of NVIDIA Jetson Nano GPUs”. In: *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. 2020, pp. 1–3.

[41] G. Oriolo, G. Ulivi, and M. Vendittelli. “Real-time map building and navigation for autonomous robots in unknown environments”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 28.3 (1998), pp. 316–333. DOI: 10.1109/3477.678626.

[42] Philip Arm et al. “Scientific exploration of challenging planetary analog environments with a team of legged robots”. In: *Science Robotics* 8.80 (2023), eade9548.

[43] Massimo Tipaldi and Bernhard Bruenjes. “Survey on fault detection, isolation, and recovery strategies in the space domain”. In: *Journal of Aerospace Information Systems* 12.2 (2015), pp. 235–256.

[44] M.L. Visinsky, J.R. Cavallaro, and I.D. Walker. “Robotic fault detection and fault tolerance: A survey”. In: *Reliability Engineering & System Safety* 46.2 (1994), pp. 139–158. ISSN: 0951-8320.

[45] Raffaella Mattone and Alessandro De Luca. “Relaxed fault detection and isolation: An application to a nonlinear case study”. In: *Automatica* 42.1 (2006), pp. 109–116.

[46] M.L. McIntyre et al. “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.

[47] F Baghernezhad and Khashayar Khorasani. “Computationally intelligent strategies for robust fault detection, isolation, and identification of mobile robots”. In: *Neurocomputing* 171 (2016), pp. 335–346.

[48] Eliahu Khalastchi and Meir Kalech. “On fault detection and diagnosis in robotic systems”. In: *ACM Computing Surveys (CSUR)* 51.1 (2018), pp. 1–24.

[49] Alessandro Marino, Francesco Pierri, and Filippo Arrichiello. “Distributed Fault Detection Isolation and Accommodation for Homogeneous Networked Discrete-Time Linear Systems”. In: *IEEE Transactions on Automatic Control* 62.9 (2017), pp. 4840–4847. DOI: 10.1109/TAC.2017.2694556.

[50] Sandra Hayden, Adam Sweet, and Scott Christa. “Livingstone model-based diagnosis of Earth Observing One”. In: *AIAA 1st Intelligent Systems Technical Conference*. 2004, p. 6225.

[51] Ryan Mackey 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.

[52] Miroslav Šimandl and Ivo Punčochař. “Active fault detection and control: Unified formulation and optimal design”. In: *Automatica* 45.9 (2009), pp. 2052–2059.

[53] M. Sampath, S. Lafortune, and D. Teneketzis. “Active diagnosis of discrete-event systems”. In: *IEEE Transactions on Automatic Control* 43.7 (1998), pp. 908–929.

[54] Elodie Chanthery, Yannick Pencole, and Nicolas 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.

[55] Elodie Chanthery et al. “Applying active diagnosis to space systems by on-board control procedures”. In: *IEEE Transactions on Aerospace and Electronic Systems* 55.5 (2019), pp. 2568–2580.

[56] Lars Blackmore and Brian 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.

[57] Joel A. Paulson et al. “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.

[58] Joseph K. Scott et al. “Input design for guaranteed fault diagnosis using zonotopes”. In: *Automatica* 50.6 (2014), pp. 1580–1589.

[59] Stephen L Campbell and Ramine Nikoukhah. *Auxiliary signal design for failure detection*. Vol. 11. Princeton University Press, 2015.

[60] S.L. Campbell, K.G. Horton, and R. Nikoukhah. “Auxiliary signal design for rapid multi-model identification using optimization”. In: *Automatica* 38.8 (2002), pp. 1313–1325.

[61] SL Campbell et al. “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.

[62] Josh Bongard, Victor Zykov, and Hod Lipson. “Resilient machines through continuous self-modeling”. In: *Science* 314.5802 (2006), pp. 1118–1121.

[63] Kaiyu Hang et al. “Manipulation for self-identification, and self-identification for better manipulation”. In: *Science Robotics* 6.54 (2021), eabe1321.

[64] Boyuan Chen et al. “Fully body visual self-modeling of robot morphologies”. In: *Science Robotics* 7.68 (2022), eabn1944.

[65] Kathleen A. Svendsen and Mae 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.

[66] Guy Shani, Joelle Pineau, and Robert Kaplow. “A survey of point-based POMDP solvers”. In: *Autonomous Agents and Multi-Agent Systems* 27.1 (2013), pp. 1–51.

[67] Cameron Browne et al. “A Survey of Monte Carlo Tree Search Methods”. In: *IEEE Trans. Comput. Intell. AI Games* 4.1 (2012), pp. 1–43.

[68] G Friedrich and I Obreja. “Model-based decision tree generation for diagnosis and measurement selection”. In: *IFAC Proceedings Volumes* 22.19 (1989), pp. 109–115.

[69] Luca Console, Claudia Picardi, and D Theseider Dupre. “Temporal decision trees: Model-based diagnosis of dynamic systems on-board”. In: *Journal of Artificial Intelligence Research* 19 (2003), pp. 469–512.

[70] Pascal Poupart et al. “Approximate linear programming for constrained partially observable Markov decision processes”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.

[71] Dongho Kim et al. “Point-based value iteration for constrained POMDPs”. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. Vol. 11. 2011, pp. 1968–1974.

[72] Kyle Hollins Wray and Kenneth Czuprynski. “Scalable Gradient Ascent for Controllers in Constrained POMDPs”. In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 9085–9091.

[73] Aditya Undurti and Jonathan P. How. “An online algorithm for constrained POMDPs”. In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 3966–3973.

[74] Arec Jamgochian, Anthony Corso, and Mykel 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. 1. 2023, pp. 198–202.

[75] Sylvie Thiebaux, Brian Williams, et al. “RAO*: An algorithm for chance-constrained POMDP’s”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[76] Majid Khonji, Ashkan Jasour, and Brian C Williams. “Approximability of Constant-horizon Constrained POMDP.” In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2019, pp. 5583–5590.

[77] Michael P. Vitus and Claire 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.

[78] Vadim Indelman, Luca Carlone, and Frank Dellaert. “Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments”. In: *The International Journal of Robotics Research* 34.7 (2015), pp. 849–882.

[79] Danijar Hafner et al. “Mastering atari with discrete world models”. In: *arXiv preprint arXiv:2010.02193* (2020).

[80] Rafael Rafailov et al. “Offline reinforcement learning from images with latent space models”. In: *Proceedings of Machine Learning Research*. 2021, pp. 1154–1168.

[81] Dibya Ghosh et al. “Offline RL Policies Should Be Trained to be Adaptive”. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. 17–23 Jul 2022, pp. 7513–7530.

[82] Ruijie He, Emma Brunskill, and Nicholas Roy. “PUMA: Planning under uncertainty with macro-actions”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 24. 1. 2010, pp. 1089–1095.

[83] Hang Ma and Joelle Pineau. “Information gathering and reward exploitation of subgoals for POMDPs”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.

[84] Matthijs TJ Spaan, Tiago S Veiga, and Pedro U Lima. “Decision-theoretic planning under uncertainty with information rewards for active cooperative perception”. In: *Autonomous Agents and Multi-Agent Systems* 29 (2015), pp. 1157–1185.

[85] Louis Dressel and Mykel Kochenderfer. “Efficient decision-theoretic target localization”. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 27. 2017, pp. 70–78.

[86] Juan Carlos Saborio and Joachim 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.

[87] Anushri Dixit, Mohamadreza Ahmadi, and Joel W. Burdick. “Risk-Sensitive Motion Planning using Entropic Value-at-Risk”. In: *2021 European Control Conference (ECC)*. 2021, pp. 1726–1732.

[88] Richard H Byrd, Jean Charles Gilbert, and Jorge Nocedal. “A trust region method based on interior point techniques for nonlinear programming”. In: *Mathematical programming* 89 (2000), pp. 149–185.

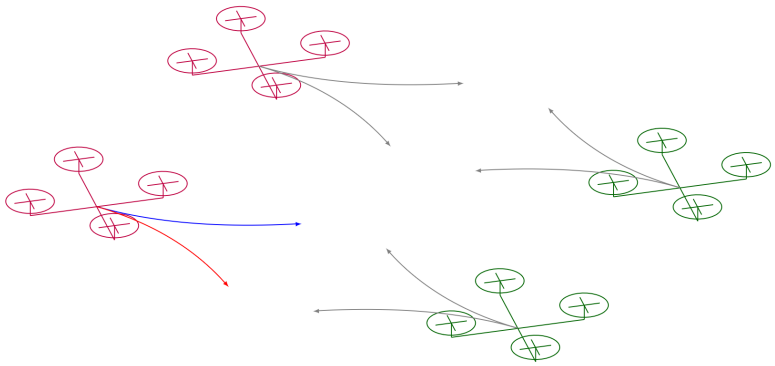
[89] Hiroyasu Tsukamoto et al. “CaRT: Certified Safety and Robust Tracking in Learning-Based Motion Planning for Multi-Agent Systems”. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*. 2023, pp. 2910–2917.

[90] Matti Vahs and Jana Tumova. “Risk-aware Control for Robots with Non-Gaussian Belief Spaces”. In: *arXiv preprint arXiv:2309.12857* (2023).

[91] Zakariya Laouar et al. “Feasibility-Guided Safety-Aware Model Predictive Control for Jump Markov Linear Systems”. In: *arXiv preprint arXiv:2310.14116* (2023).

Chapter 4

MCTS FOR MULTI-AGENT GAMES



How to construct trees and search with multiple rational agents?

This chapter is based on the publications:

> Benjamin Rivière et al. "Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environments". In: *IEEE Robotics Automation Letters* 6.4 (2021), pp. 6868–6875. DOI: 10.1109/LRA.2021.3096758.

4.1 Motivation

Multi-agent interactions in non-cooperative environments are ubiquitous in robotic applications such as self-driving, space exploration, urban air mobility, and human-robot collaboration. Planning in these settings requires a prediction model of the other agents, which can be generated through a game theoretic framework.

Recently, the success of AlphaZero [2] at the game of Go has popularized a self-improving machine learning algorithm: bias a Monte Carlo Tree Search with value and policy neural networks, use the tree statistics to train the networks with supervised learning and then iterate over these two steps to improve the policy and value networks over time. Although this approach has been demonstrated for classical artificial intelligence problems like chess or Go, its application applications in multi-robot domains, with different assumptions such as continuous state-action space, decentralized evaluation, partial information, and limited computational resources, requires new algorithm development and hardware validation.

4.2 Problem Statement

We consider N agents in the index set, $\mathcal{I} = \{0, 1, \ldots, N\}$. For each agent i, we specify the dynamics and reward function:

$$x_{k+1}^i = F^i(x_k^i, u_{k+1}^i) \tag{4.1}$$

$$r_k^i = R^i(x_k, u_k) \tag{4.2}$$

where x^i and u^i is the ith agent state and action and x, u are the global state and action of all agents.

Given an initial state, the ith agent's policy π^i, and the policy of all other agents π^{-i}, we compute the value:

$$V(x_0, \pi^i, \pi^{-i}) = \sum_{k=1}^{K} \gamma^k r_k^i + \gamma^H D^i(x_K) \quad \text{s.t.} \quad (4.1), (4.2) \tag{4.3}$$

where D^i is a boundary condition.

The Nash Equillibrium (NE) is the notion of optimality in non-cooperative games [3]. A set of policies $\{\pi^{1,*}, \ldots, \pi^{N,*}\}$ are NE if no agent has incentive to change its policy:

$$V^*(x_0, \pi^{i,*}, \pi^{-i,*}) \geq V(x_0, \pi^i, \pi^{-i,*}) \quad \forall i \in \mathcal{I} \tag{4.4}$$

For notation compactness, we sometimes denote the problem data with $\mathcal{M}$:

$$\mathcal{M} = \langle \{X^i\}_{i \in \mathcal{I}}, \{U\}_{i \in \mathcal{I}}, \{F^i\}_{i \in \mathcal{I}}, \{R^i\}_{i \in \mathcal{I}}, \{D^i\}_{i \in \mathcal{I}}, K, \gamma \rangle \tag{4.5}$$

4.3 Neural Tree Expansion Algorithm

We present our method, which is composed of a tree search algorithm, an offline training algorithm, and a global-to-local supervised learning technique. We summarize our policy notation: optimal policy π^*, heuristic policy $\tilde{\pi}$, offline tree policy π^e, and real-time tree policy π^l. The value function follows similar notation: optimal policy V^*, heuristic value $\tilde{V}$, etc..

Neural Tree Search

We use the conventional Monte Carlo Tree Search (MCTS) algorithm from [4], modified with our neural heuristics, shown in Algorithm 3. The tree search is used in real-time deployment and offline data generation.

MCTS begins at some start state x and grows the tree until its computational budget is exhausted, measured by the number of nodes in the tree, L. Each node in the

tree is a state, x, each edge is an action u, and each child is the new state after propagating the dynamics. Each node stores: the state vector, $x(q)$, the action-to-node $u(q)$, the reward-to-node $r(q)$, the total value $V(q)$, the number of visits to the node, $N(q)$ and its children set, $C(q)$. The growth iteration in the main function, Search, has four steps: (i) Select, selects a node to balance exploration of space and exploitation of rewards (ii) Expand, creates a child node by forward propagating the selected node with an action either constructed by the neural network or by random sampling, (iii) DefaultPolicy collects terminal reward statistics by rolling out a simulated state trajectory from the new node, and (iv) Backpropagate updates the number of visits and cumulative reward up the tree. Each depth in the tree corresponds to the turn of an agent and their action is predicted by selecting the best node for their cost function. The final action returned by the search is the child of the root node with the most visits.

The changes we propose from standard MCTS are the integration of a policy and value neural networks heuristics. First, the policy network modifies the Expand function by, with some frequency β_π, generating actions from a learned probability distribution, rather than sampling uniformly from the action space. Second, the value network changes the DefaultPolicy function by, with some frequency β_V, replacing a sample of the value of a random rollout with a learned prediction of the value. The details of training these heuristics are presented later.

Both the offline policy π^e and real-time tree policy π^l use this same algorithm for tree search, with two key differences: First, the number of simulations in the tree L, is much larger for the offline policy than in the real-time setting. The motivation for this change is that the real-time setting has a limited computational budget and, although the offline setting does not have this restriction, it has the additional requirement of producing high quality data to train the neural network heuristics. Second, the offline policy is centralized, i.e. it outputs a joint-space action for all agents, whereas the real-time policy is decentralized, i.e. it is run in parallel for each agent, and outputs an action for a single agent. We require a decentralized real-time deployment to ensure operation in the case of agent failure and to provide high scalability in the number of agents. We use a centralized training phase to provide coordinated maneuvers in the dataset, this

Algorithm 3: Neural Tree Expansion

def Search$(x, \tilde{\pi}, \tilde{V})$**:**
  $q \leftarrow \text{Node}(x, \text{None})$;
  for $l = 1, \ldots, L$ **do**
    $q_l \leftarrow \text{Expand}(\text{Select}(q_0), \tilde{\pi})$;
    $v \leftarrow \text{DefaultPolicy}(x(q_l), \tilde{V})$;
    Backpropagate(q_l, v) ;
  $q^* = \arg\max_{q' \in C(q_0)} V(q'))$;
  return $V(q^*)u(q^*)$;
def Select(q)**:**
  return $\arg\max_{q' \in q(C)} \frac{q'(V)}{q'(N)} + c\sqrt{\frac{\log(q(N))}{q'(N)}}$;
def Expand$(q, \tilde{\pi})$**:**
  $\alpha \sim \mathbb{U}(0, 1)$;
  if $\alpha < \beta_\pi$ **then**
    $u \leftarrow [u^1, \ldots, u^{|\mathcal{I}|}], \quad u^i \sim \tilde{\pi}^i(x), \quad \forall i \in \mathcal{I}$
  else
    $u \leftarrow [u^1, \ldots, u^{|\mathcal{I}|}], \quad u^i \sim \mathbb{U}(U^i), \quad \forall i \in \mathcal{I}$;
  $q' \leftarrow \text{Node}(F(x, u), q)$;
  return q' ;
def DefaultPolicy$(x, \tilde{V})$**:**
  $\alpha \sim \mathbb{U}(0, 1)$;
  if $\alpha < \beta_V$ **then**
    $v \sim \tilde{V}(x)$;
  else
    $v = 0$;
    while x is not terminal **do**
      $u \leftarrow [u^1, \ldots, u^{|\mathcal{I}|}], \quad u^i \sim \mathbb{U}(U^i), \quad \forall i \in \mathcal{I}$;
      $x \leftarrow F(x, u)$;
      $v \mathrel{+}= R(x, u)$;
  return v ;
def Backpropagate(q, v)**:**
  while q is valid **do**
    $N(q) \leftarrow N(q) + 1$;
    $V(q) \leftarrow V(q) + v$;
    $q \leftarrow P(q)$;

Offline Training

The offline algorithm trains the neural network heuristics, $\tilde{\pi}$ and $\tilde{V}$, and its pseudocode is presented in Algorithm 4.

The desired behavior of a single iteration of the offline training loop is to improve the policy heuristic by decreasing the distance between the heuristic policy network and the optimal policy function:

$$\|\tilde{\pi}_{k+1} - \pi^*\| \leq \|\tilde{\pi}_k - \pi^*\| \tag{4.6}$$

where π^* is the unknown optimal policy function, $\tilde{\pi}_k$ is the policy network we train, and k is the iteration of the training loop.

The strategy for satisfying the policy improvement condition (4.6) is in two steps: (i) create a dataset of tree search examples where each demonstration data is closer to optimal than the heuristic:

$$\mathcal{D}_k^{\pi} = \{x, \pi_k^e(x)\} \quad \text{s.t.} \quad \|\pi^*(x) - \pi_k^e(x)\| \leq \|\pi^*(x) - \tilde{\pi}_k(x)\| \tag{4.7}$$

and, (ii) supervised learning to train new heuristics that imitate the improved demonstration data:

$$\tilde{\pi}_{k+1} = \arg\min_{\tilde{\pi}} \sum_{(x,\pi^e(x))\in\mathcal{D}} \|\tilde{\pi}(x) - \pi^e(x)\| \tag{4.8}$$

This is a training loop empirically popularized by [5] and theoretically analyzed in [6]. The process for the value learning is analogous. Both processes are summarized in the Algorithm 4, and $\mathcal{L}^{\pi}$ are the respective loss functions.

Policy Network: The tree search is guided by N policy heuristic functions for each robot i. These functions maps state observations to the action distribution for a single robot, and they are used in the tree search to create edges to children nodes. The desired behavior of the policy network is to generate individual robot actions with a high probability of being near-optimal expansions given the current observation, i.e. generate edges to nodes with a high number of visits in the expert search. The training of a decentralized policy from centralized data is a global-to-local learning technique [7].

Our implementation of the policy function is different from the AlphaZero methods; whereas AlphaZero's policy *selects* from a set of existing discrete set, our expansion *generates* an action from a learned probability distribution. This additional step is necessary to search in the robot's continuous action space.

Algorithm 4: Offline Training

def Offline Training($\mathcal{M}$)**:**
  $\tilde{\pi}_0, \tilde{V}_0 = \text{None}, \text{None}$
  for $k = 0, \ldots, K$ **do**
    $\mathcal{D}_k^{\pi}, \mathcal{D}_k^{V} = \{\}, \{\}$
    for $j = 0, \ldots, |D|$ **do**
      $x_j \sim X$
      $V^e(x_j), u^e(x_j) = \pi^e(x_j, \tilde{\pi}_k, \tilde{V}_k), \mathcal{M})$
      $\mathcal{D}_k^{\pi}.\text{add}(x_j, u^e(x_j))$
      $\mathcal{D}_k^{V}.\text{add}(x_j, V^e(x_j))$
    end
    for $i \in \mathcal{I}$ **do**
      $\tilde{\pi}_{k+1}^i = \arg\min_{\tilde{\pi}} \sum_{(x,\pi^e(x))\in\mathcal{D}_k^V} \mathcal{L}^{\pi}(\tilde{\pi}(x), \pi^e(x))$
    end
    $\tilde{V}_{k+1} = \arg\min_{\tilde{V}} \sum_{(x,V^e(x))\in\mathcal{D}_k^V} \|\tilde{V}(x) - V^e(x)\|$
  end

The dataset for each robot i's policy network is composed of observation action pairs. The action label is calculated by querying the expert at some state, extracting the root node's child distribution, and calculating the average of the action distribution weighted by the relative number of visits:

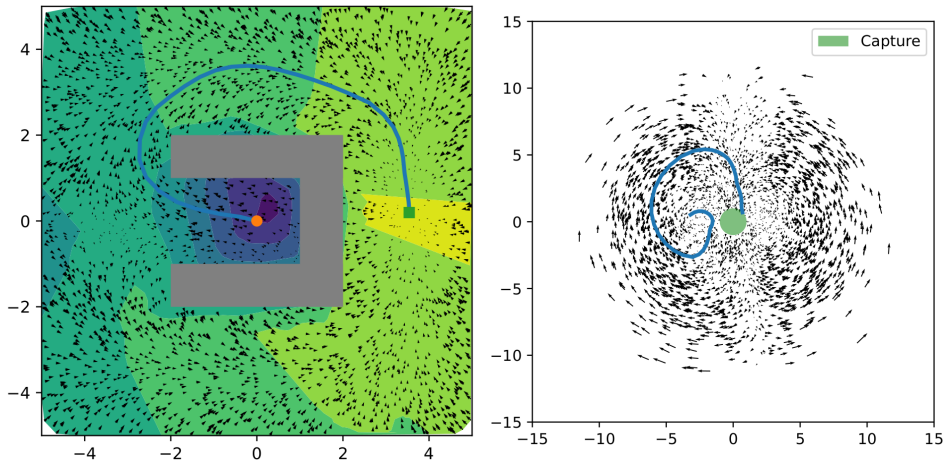
$$u_l^i = \sum_{q' \in C(q_0)} \frac{N(q')}{N(q_0)} A^i(q_0, q') \tag{4.9}$$

where u_l^i is the action label, q_0 is the root node, $C(\cdot)$ is the set of child nodes, $N(\cdot)$ is the number of visits to a node, and $A^i(q_0, q')$ is the ith robot's action from root node q_0 to child node q'.

The policy network outputs the mean and variance of a multivariate Gaussian distribution An action sample $\hat{u}_l^i$ can then be computed by sampling ϵ and transforming it by the neural network's mean and variance output:

$$\hat{u}_l^i = \mu(x_l) + \sigma(x_l)\epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{4.10}$$

The input, x_l, is encoded with a DeepSet [8] feedforward architecture similar to [7] that is compatible with a variable number of neighboring robots. The maximum likelihood solution to the multivariate Gaussian problem is found by minimizing



(a) Bugtrap Problem.

(b) Homicidal Chauffeur Problem

Figure 4.1: NTE applied to the Bugtrap and Homicidal Chauffeur Problem.

the following loss function:

$$\mathcal{L}^{\pi} = \sum_{l} (u_l^i - \mu)^T \sigma^{-1} (u_l^i - \mu) + \frac{1}{2} \ln|\sigma| \tag{4.11}$$

$$\tilde{\pi}^i = \arg\min_{\tilde{\pi}^i \in \Pi^i} \sum_{x_l, u_l^i \in \mathcal{D}} \mathcal{L}(\mu(x_l), \sigma(x_l), u_l^i) \tag{4.12}$$

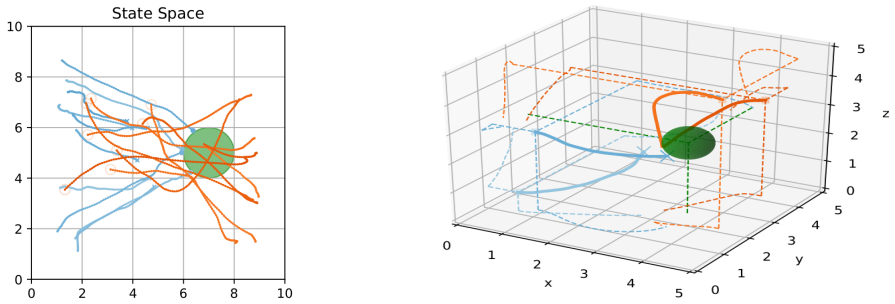
where μ and σ are generated by the neural network given x_l, and u_l^i is the target.

Value Network: The value network is used to gather reward statistics in place of a policy rollout and its desired behavior is to predict the outcome of games if they were rolled out with the current policy network. Our implementation of the value function is the same as AlphaZero methods. The value network outputs state-dependent mean and variance of a Gaussian distribution, and is trained with a similar loss function as the policy network (4.11), using a learning target of the value labels. The value is also queried from the neural network in a similar fashion (4.10) and uses a similar model architecture as the policy network, permitting variable input size of x.

4.4 Simulation Results

Bugtrap and Homicidal Chauffer

Neural Tree Expansion (NTE) can be applied to decision-making problems such as single agent motion planning and canonical differential games. Here we present two visual examples, before presenting an in-depth quantitative analysis in the next section.



(a) NTE scales to high dimensional team games for the 10 agent vs. 10 agent "Reach-Target-Avoid" with double-integrator dynamics.

(b) NTE is compatible with arbitrary dynamics, here is the "Reach-Target-Avoid" game with 3D Dubin's vehicle dynamics.

Figure 4.2: NTE applied to Reach-Target Avoid Game.

In Fig. 4.1a NTE finds the intuitive value and policy function for the "bugtrap" motion planning problem. The robot starts at the orange dot, and terminates at the green square after reaching the goal. In Fig. 4.1b The state trajectories generated by NTE approximate the primary solution and barrier surface for the "homicidal chauffeur" game [9]. The plot is shown in Isaac's reduced space with an example trajectory in blue terminating in a capture condition.

Reach Target Avoid Game with Double Integrators

A well-studied differential game is the Reach-Target-Avoid game [9]. For two teams of robots, team A gets points for robots that reach the goal region, and team B gets points for defending the goal by tagging the invading robots first. The teams are specified with index sets $\mathcal{I}_A$ and $\mathcal{I}_B$, respectively, where the union of the two teams represents all robots, $\mathcal{I}_A \cup \mathcal{I}_B = \mathcal{I}$. An example of the Reach-Target-Avoid game is shown in Fig. 4.2a, where the red robots try to tag the blue robots before the blue robots reach the green goal region. The x and o on the trajectory indicates tagged state and reached goal.

Variants and Baseline: In order to evaluate our method, we test the multiple learners and expert policies, each equipped with networks after k learning iterations. To isolate the effect of the neural expansion, we consider the $k = 0$ case for both learner and expert as an unbiased MCTS baseline solution.

As an additional baseline for the double-integrator game, we use the solution from [10]. Their work adapts the exact differential game solution for simple-motion and single-robot teams proposed in [9] to a double-integrator, multi-robot team setting. How-

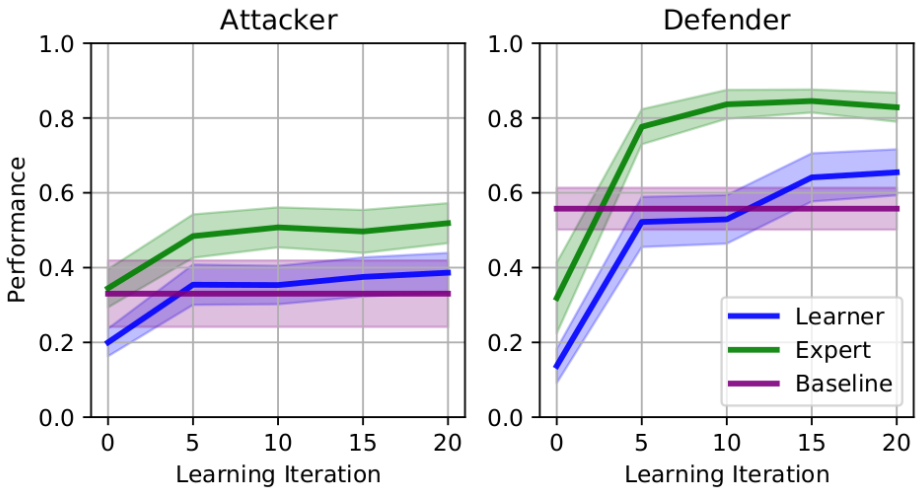
ever, their adapted solution is not exact because it assumes a constant acceleration magnitude input and relies on composition of pair-wise matching strategies. For this reason, our solution will be able to outperform this hand-crafted strategy.

Experiment: We evaluate our expert and learner by initializing 100 different initial conditions of a 3 attacker, 2 defender game in a 3 m space. Then, we rollout every combination of variants, learning iterations, and baseline for both team A and team B policies, for a total of 12 100 games. For a single game, the performance criteria for team A policies is the terminal reward and, in order to have consistency of plots (higher is better), the performance criteria for team B policies is one minus the terminal reward. An example game with a different number of agents and environment size is shown in Fig. 4.2a and its animation is provided in the supplemental video. The 10 vs. 10 game illustrates the natural scalability in number of agents of the decentralized approach and the generalizability of the neural networks, as they were only trained with data containing up to 5 robots per team.

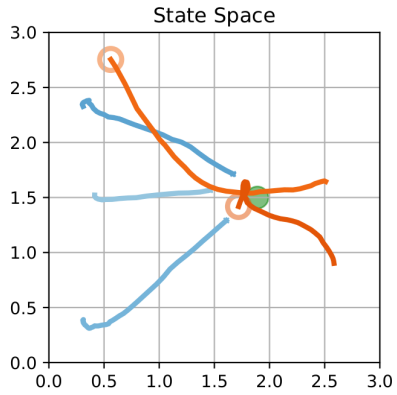
The statistical results of the 3 vs. 2 experiment are shown in Fig. 4.3a where the thick lines denote the average performance value and the shade is the performance variance. We find the expected results; for both team A and team B, the learner with no bias has the worst performance, and learner with fully trained networks surpasses the centralized and expensive unbiased expert and approaches the biased expert. The baseline attacker is about the same strength as the unbiased expert, whereas the baseline defender is much stronger than the unbiased expert. In both cases, the fully-trained biased expert and learner are able to significantly outperform the baseline.

To investigate the qualitative advantages of our method, we looked at the games where our learner defense outperformed the baseline defense and found two principal advantages: first, the learner defense sometimes demonstrated emergent coordination that is more effective than a pairwise matching strategy, e.g. one defender goes quickly to the goal to protect against greedy attacks while the other defender slowly approaches the goal to maintain its maneuverability, see Fig. 4.3b. Second, the learner attacker is sometimes able to exploit the momentum of the baseline defender and perform a dodge maneuver, e.g. the bottom left interaction in Fig. 4.3c, whereas the learner defense is robust to this behavior. These examples show the learner networks can generate sophisticated, effective maneuvers.

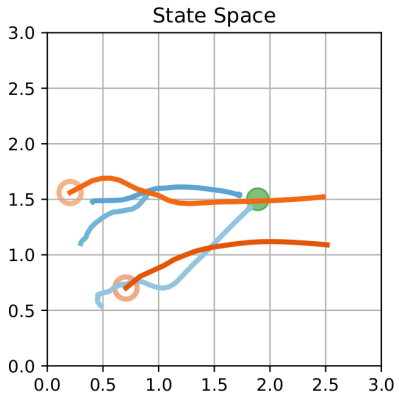
As an additional experiment, we evaluate the learner (without retraining) in an environment with static and dynamic obstacles for 100 different initial conditions; an



(a) Double-integrator game evaluation: the thick lines indicate the average performance and the shaded area is the variance over 100 games.



(b) Learner defense (orange) finds emergent cooperative strategies to defend the goal (green).



(c) Baseline defense (orange) is vulnerable to learner's offensive (blue) dodge maneuver.

Figure 4.3: Double-integrator performance and strategy examples.

example is shown in the supplementary video. In this environment, the fully trained learner outperforms the unbiased learner 0.246 ± 0.022, this value is calculated by summing the performance criteria difference across attacking and defending policies. This result demonstrates the natural compatibility of tree-based planners with safety constraints and the robustness of the performance gain in out-of-training-domain scenarios.

Reach Target Avoid Game with 3D Dubin's vehicles

As shown in Fig. 4.2, NTE can be applied to arbitrary game settings and dynamics. We evaluate the same Reach-Target-Avoid game with 3D Dubin's vehicle dynamics

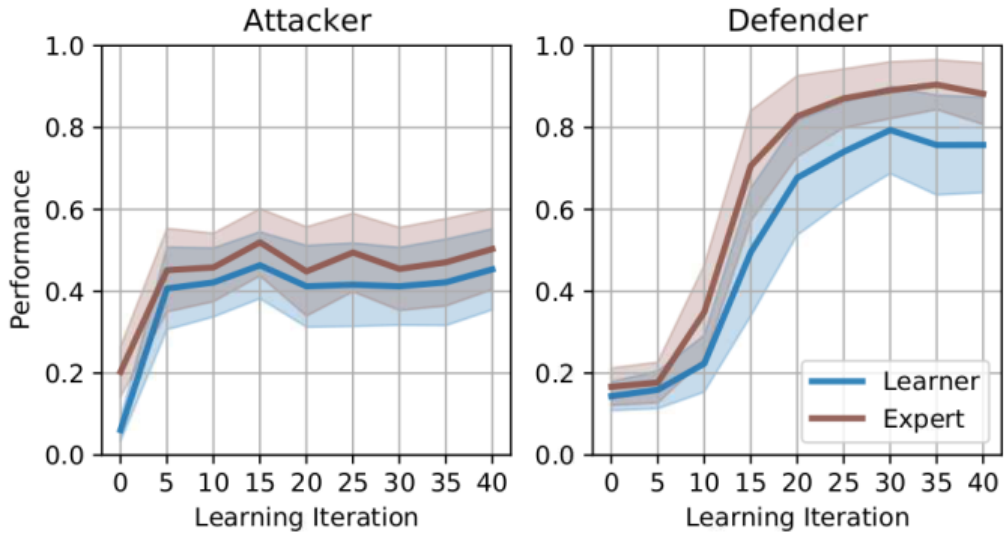


Figure 4.4: 3D Dubin's vehicle game evaluation: the thick lines indicate the average performance and the shaded area is the variance over 100 games.

as a relevant model for fixed-wing aircraft applications, shown in Fig. 4.2b. We consider the state, action, and dynamics: $x_t = [x_t, y_t, z_t, \psi_t, \gamma_t, \phi_t, v_t]^T$, $u_t = [\dot{\gamma}_t, \dot{\phi}_t, \dot{v}_t]^T$

$$x_{t+1} = F(x_t, u_t) = x_t + \begin{bmatrix} v_t \cos(\gamma_t)\sin(\psi_t) \\ v_t \cos(\gamma_t)\cos(\psi_t) \\ -v_t \sin(\gamma_t) \\ \frac{g}{v_t}\tan(\phi_t) \\ u_t \end{bmatrix} \Delta_t \tag{4.13}$$

where x, y, z are inertial position, v is speed, ψ is the heading angle, γ is the flight path angle, and ϕ is the bank angle and g is the gravitational acceleration. The game is bounded to $\underline{x} = \overline{x}$ =5 m with a maximum linear acceleration of 2.0 m/s^2 and maximum angular rates of 36 deg/s, and g is set to 0.98 m/s^2 to scale to our game length scale.

We initialize 2 attacker, 2 defender games for 100 different initial conditions in a 5 m region and test the policy variants, without an external baseline, for a total of 81 000 games. The performance results are shown in Fig. 4.4, where we see the same trend that the learner and expert policies improve over learning iterations. In addition, the biased learner's performance quickly surpasses the unbiased expert ($k = 0$).

4.5 Hardware Result

To test our algorithm in practice, we fly in a motion capture space, where each robot (CrazyFlie 2.x, see Fig. 4.5) is equipped with a single marker, and we use

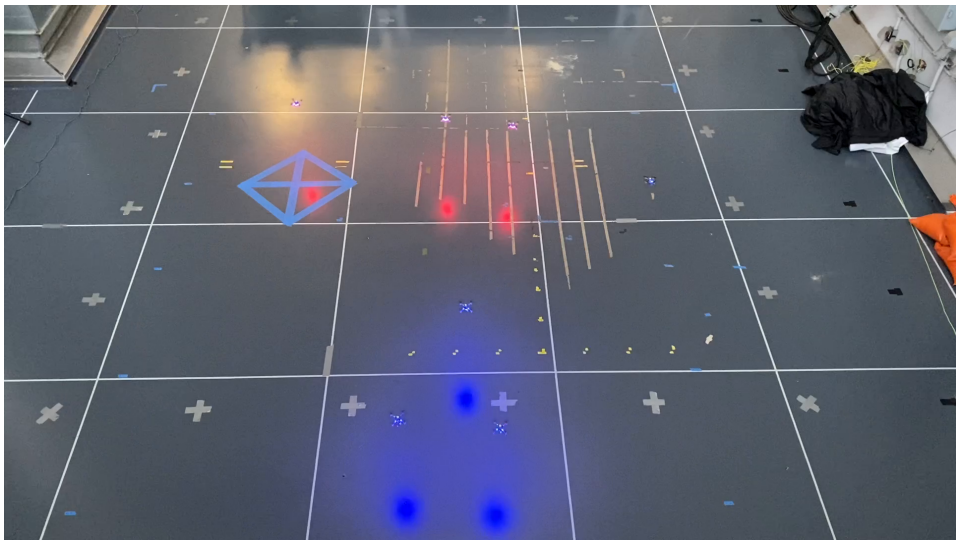


Figure 4.5: NTE Swarm Game Hardware Demonstration

the Crazyswarm [11] for tracking and scripting. The centralized system simulates distributed operation by collecting the full state, computing local observations and local policies, and broadcasting only the output of each robot's learner policy. For a given double-integrator policy, we evaluate the learner to compute an action, forward-propagate double-integrator dynamics, and track the resulting position and velocity set-point using a nonlinear controller for full quadrotor dynamics. Planning in a lower-dimensional double-integrator state and then tracking the full system is enabled by the timescale separation of position and attitude dynamics of quadrotors.

We evaluate the double-integrator learner for up to 3 attacker, 2 defender games in an aerial swarm flight demonstration. We show the results of the experiments in our supplemental video. We use the same parameters as in simulation in Sec. 4.4. Our learner evaluation takes an average of 11 ms with a standard deviation of 6 ms, with each robot policy process running in parallel on an Intel(R) Core(TM) i7-8665U. By comparison, the biased expert takes 329±144 ms to execute and the unbiased expert takes 277±260 ms. Our computational tests show that the learner has a significant ($\approx$ 25 times) computational advantage over the baseline unbiased expert. Our physical demonstration shows that our learner is robust to the gap between simulation and real world and can run in real-time on off-the-shelf hardware.

4.6 Related Work

Our work relates to multiple communities: planning, machine learning, and game theory.

Planning, or sequential decision-making, problems can be solved in an online setting with Monte Carlo Tree Search (MCTS) [12]. MCTS searches through the large decision-making space by rolling out simulated trajectories and biasing the tree growth towards areas of high reward [4]. MCTS was first popularized by the Upper Confidence Bound for Trees algorithm [13] that uses a discrete-action, multi-armed bandit solution to balance exploration and exploitation in node selection. Recent work uses a non-stationary bandit analysis to propose a polynomial, rather than logarithmic, exploration term [14]. As an anytime algorithm, the space and time complexity of MCTS is user-determined by the desired number of simulations. Recent finite sample complexity results of MCTS [14, 15] show the error in root node value estimation converges at a rate of the order $n^{-1/2}$ where n is the number of simulations.

Application of MCTS to a dynamically-constrained robot planning setting requires extending the theoretical foundations to a continuous state and action space. In general, the recent advances in this area answer two questions: i) how to select an action from the continuous space, and ii) how to determine when a node is fully expanded. Regarding the former question, some solutions select an action using the extension of the multi-armed bandit in continuous domains [16], and other approaches sample uniformly from the continuous space. In contrast, our approach uses a policy network to learn a distribution from which we sample to generate actions. Regarding the latter question, a popular method to determine whether a node is expanded is to use progressive widening and variants; we adapt one such method, the Polynomial Upper Continuous Trees (PUCT) algorithm [17]. Despite the advance in theory for continuous action spaces, there have been relatively few studies of biasing continuous MCTS with deep neural networks [18].

The key idea of AlphaZero [2] is using MCTS as a policy improvement operator; i.e. given a policy neural network to guide MCTS, the resulting search produces an action closer to the optimal solution than that generated by the neural network. Then, the neural network is trained with supervised learning to imitate the superior MCTS policy, matching the quality of the network to that of MCTS in the training domain. By iterating over these two steps, the model improves over time. The first theoretical analysis of this powerful method is recently shown for single-agent

discrete action space problems [14]. In comparison, our method is applied to a continuous state-action, multi-agent setting. Whereas AlphaZero methods use the policy network to bias the node selection process, i.e. given a list of actions, select the best one, our policy network is an action generator for the expansion process to create edges to children, i.e. given a state, generate an action. A neural expansion operator has previously been explored in motion planning [19], but not decision-making. In addition, our method's supervised learning step is closer to imitation learning, as used in DAgger [20], because the learner benefits from an adaptive dataset generation of using self-play to query from an expert.

Although the AlphaZero methods use a form of supervised learning to train the networks, they can be classified as a reinforcement learning method because the networks are trained without a pre-existing labelled dataset. Policy gradient [21] is a conventional reinforcement learning solution and there are many recent advances in this area [22]. Adding an underlying tree structure to deep reinforcement learning provides a higher degree of interpretability and a more stable learning process, enabled by MCTS's policy improvement property.

In contrast to data-driven methods, traditional analytical solutions can be studied and derived through differential game theory. The game we study, Reach-Target-Avoid, was first introduced and solved for simple-motion, 1 vs. 1 systems [9]. Later, multi-robot, single-integrator solutions have been proposed [23, 24]. Solutions considering multi-robots with non-trivial dynamics, such as the double-integrator [10], are an active area of research. Shepherding, herding, and perimeter defense are variants of the Reach-Target-Avoid game and are also active areas of research [25–28].

BIBLIOGRAPHY

[1] Benjamin Rivière, Wolfgang Hönig, Matthew Anderson, and Soon-Jo Chung. "Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environments". In: *IEEE Robotics Automation Letters* 6.4 (2021), pp. 6868–6875. DOI: 10.1109/LRA.2021.3096758.

[2] David Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (2018), pp. 1140–1144.

[3] Tamer Basar and Georges Zaccour. *Handbook of dynamic game theory*. Springer, 2018.

[4] Cameron Browne et al. "A Survey of Monte Carlo Tree Search Methods". In: *IEEE Trans. Comput. Intell. AI Games* 4.1 (2012), pp. 1–43.

[5] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (2017), pp. 354–359. URL: https://doi.org/10.1038/nature24270.

[6] Devavrat Shah, Qiaomin Xie, and Zhi Xu. "Non-asymptotic analysis of monte carlo tree search". In: *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*. 2020, pp. 31–32.

[7] Benjamin Rivière et al. "GLAS: Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning With End-to-End Learning". In: *IEEE Robot. Autom. Lett.* 5.3 (May 2020), pp. 4249–4256.

[8] Manzil Zaheer et al. "Deep Sets". In: *Neural Inf. Process. Syst.* 2017, pp. 3391–3401.

[9] Rufus Isaacs. *Differential games; a mathematical theory with applications to warfare and pursuit, control and optimization*. Wiley, 1965.

[10] Mitchell Coon and Dimitra Panagou. "Control Strategies for Multiplayer Target-Attacker-Defender Differential Games with Double Integrator Dynamics". In: *IEEE 56th Annual Conf. on Decis. and Control*. 2017 IEEE 56th Annual Conference on Decision and Control (CDC). 2017, pp. 1496–1502.

[11] James A. Preiss* et al. "Crazyswarm: A large nano-quadcopter swarm". In: *Proc. IEEE Int. Conf. Robot. Autom.* 2017, pp. 3299–3304. URL: https://doi.org/10.1109/ICRA.2017.7989376.

[12] Mykel J. Kochenderfer et al. *Decision Making Under Uncertainty: Theory and Application*. 1st Ed. The MIT Press, 2015.

[13] Levente Kocsis and Csaba Szepesvári. "Bandit Based Monte-Carlo Planning". In: *Eur. Conf. Mach. Learn.* Vol. 4212. Springer, 2006.

[14] Devavrat Shah, Qiaomin Xie, and Zhi Xu. “Non-Asymptotic Analysis of Monte Carlo Tree Search”. In: *SIGMETRICS (Abstracts)*. ACM, 2020, pp. 31–32.

[15] W. Mao et al. “POLY-HOOT: Monte-Carlo Planning in Continuous Space MDPs with Non-Asymptotic Analysis”. In: *Neural Inf. Process. Syst.* 2020, pp. 1–11.

[16] Christopher R. Mansley, Ari Weinstein, and Michael L. Littman. “Sample-Based Planning for Continuous Action Markov Decision Processes, and Scheduling”. In: *Int. Conf. on Autom. Planning and Scheduling*. 2011, pp. 335–338.

[17] David Auger, Adrien Couëtoux, and Olivier Teytaud. “Continuous Upper Confidence Trees with Polynomial Exploration – Consistency”. In: *Eur. Conf. Mach. Learn.* Vol. 8188. Springer, 2013.

[18] Thomas M. Moerland et al. “A0C: Alpha Zero in Continuous Action Space”. In: *Eur. Workshop on Reinforcement Learn. 14*. 2018, pp. 1–10.

[19] Binghong Chen et al. “Learning to Plan in High Dimensions via Neural Exploration-Exploitation Trees”. In: *Int. Conf. on Learn. Repres.* 2020.

[20] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning”. In: *Proc. Int. Conf. Artif. Intell. and Statist.* 2011.

[21] Richard S. Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Neural Inf. Process. Syst.* 1999.

[22] Manish Prajapat et al. “Competitive Policy Optimization”. In: *Conf. on Uncertainty in Artif. Intell.* 2021.

[23] Eloy Garcia, David W. Casbeer, and Meir Pachter. “Optimal Strategies for a Class of Multi-Player Reach-Avoid Differential Games in 3D Space”. In: *IEEE Robot. Autom. Lett.* 5.3 (2020), pp. 4257–4264.

[24] Rui Yan et al. “Matching-Based Capture Strategies for 3D Heterogeneous Multiplayer Reach-Avoid Differential Games”. In: *CoRR* (2019).

[25] Aditya A Paranjape et al. “Robotic herding of a flock of birds using an unmanned aerial vehicle”. In: *IEEE Trans. Robot.* 34.4 (2018), pp. 901–915.

[26] Junyan Hu et al. “Occlusion-Based Coordination Protocol Design for Autonomous Robotic Shepherding Tasks”. In: *IEEE Trans. on Cogn. and Develop. Syst.* (2020), pp. 1–1.

[27] Simone Nardi, Federico Mazzitelli, and Lucia Pallottino. “A Game Theoretic Robotic Team Coordination Protocol For Intruder Herding”. In: *IEEE Robot. Autom. Lett.* 3.4 (2018), pp. 4124–4131.

[28] Daigo Shishika, James Paulos, and Vijay Kumar. “Cooperative Team Strategies for Multi-Player Perimeter-Defense Games”. In: *IEEE Robot. Autom. Lett.* 5.2 (2020), pp. 2738–2745.

Chapter 5

GLOBAL-TO-LOCAL LEARNING

This chapter is based on the publications:

> Benjamin Rivière et al. "GLAS: Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning With End-to-End Learning". In: *IEEE Robotics Automation Letters* 5.3 (2020), pp. 4249–4256. DOI: 10.1109/LRA.2020.2994035.

5.1 Motivation

Teams of robots that are capable of navigating in dynamic and occluded environments are important for applications in next generation factories, urban search and rescue, and formation flying in cluttered environments or in space. Current centralized approaches can plan such motions with completeness guarantees, but require full state information not available to robots on-board, and are too computationally expensive to run in real-time. Distributed approaches instead use local decoupled optimization, but often cause robots to get trapped in local minima in cluttered environments. Our approach, GLAS, bridges this gap by using a global planner offline to learn a decentralized policy that can run efficiently online. We can thus automatically synthesize an efficient policy that avoids getting trapped in many cases. Unlike other learning-based methods for motion planning, GLAS operates in continuous state space with a time-varying number of neighbors and generates provably safe, dynamically-coupled policies. We demonstrate in simulation that our policy achieves significantly higher success rates compared to ORCA, a state-of-the-art decentralized approach for single integrator dynamics. We also extend our approach to double integrator dynamics, and demonstrate that our synthesized policies work well on a team of quadrotors with low-end microcontrollers.

5.2 Problem Statement

Let $\mathcal{I}$ denote the set of N robots, $G = \{g^1, \ldots, g^N\}$ denote their respective goal states, $x_0 = \{x_0^1, \ldots, x_0^N\}$ denote their respective start states, and Ω denote the set of m static obstacles. At time t, each robot i makes a local observation, o^i, uses it to formulate an action, u^i, and updates its state, x^i, according to the dynamical model. Our goal is to find a controller, $u : O \to U$ that synthesizes actions from local

observations through:

$$o^i = o(i, x), \quad u^i = u(o^i), \quad \forall i, t \tag{5.1}$$

to approximate the solution to the optimal control problem:

$$\begin{aligned} u^* = \underset{\{u^i | \forall i,t\}}{\arg\min}\, c(x, u) \qquad & \text{s.t.} \\ \dot{x}^i = F(x^i, u^i) \qquad & \forall i, t \\ x^i(0) = x_0^i,\ x^i(t_f) = g^i,\ x \in X_s \qquad & \forall i \\ \|u^i\|_2 \leq u_{\max} \qquad & \forall i, t \end{aligned} \tag{5.2}$$

where O, U, and X are the observation, action, and state space, h is the local observation model, c is some cost function, F is the dynamical model, $X_s \subset X$ is the safe set capturing safety of all robots, t_f is the time of the simulation, and $u_{\max}$ is the maximum control bound.

Dynamical Model: We consider 2d single and double integrator systems. For each agent, the single integrator state and action are position and velocity vectors in $\mathbb{R}^2$, respectively. For each agent, the double integrator state is a stacked position and velocity vector in $\mathbb{R}^2$, and the action is a vector of accelerations in $\mathbb{R}^2$. We use a 2d workspace, but our algorithm and analysis is also applicable to a 3d workspace. The dynamics of the i^{th} robot for single and double integrator systems are:

$$\dot{x}^i = \dot{p}^i = u^i \quad \text{and} \quad \dot{x}^i = \begin{bmatrix} \dot{p}^i \\ \dot{v}^i \end{bmatrix} = \begin{bmatrix} v^i \\ u^i \end{bmatrix}, \tag{5.3}$$

respectively, where p^i and v^i denote position and velocity.

Observation Model: We are primarily focused on studying the transition from global to local, which is defined via an observation model, $h : \mathcal{I} \times X \to O$. An observation is:

$$o^i = \left[e^{ii}, \{x^{ij}\}_{j \in \mathcal{N}_\mathcal{I}^i}, \{x^{ij}\}_{j \in \mathcal{N}_\Omega^i} \right], \tag{5.4}$$

where $e^{ii} = g^i - x^i$ and $\mathcal{N}_\mathcal{I}^i$, $\mathcal{N}_\Omega^i$ denote the neighboring set of robots and obstacles, respectively. These sets are defined by the observation radius, r_{sense}, e.g.,

$$\mathcal{N}_\mathcal{I}^i = \{ j \in \mathcal{I} \mid \|p^{ij}\|_2 \leq r_{\text{sense}} \}. \tag{5.5}$$

We encode two different neighbor sets because we input robots and obstacles through respective sub-networks of our neural network architecture in order to generate heterogeneous behavior in reaction to different neighbor types. We denote the union of the neighboring sets as $\mathcal{N}^i$.

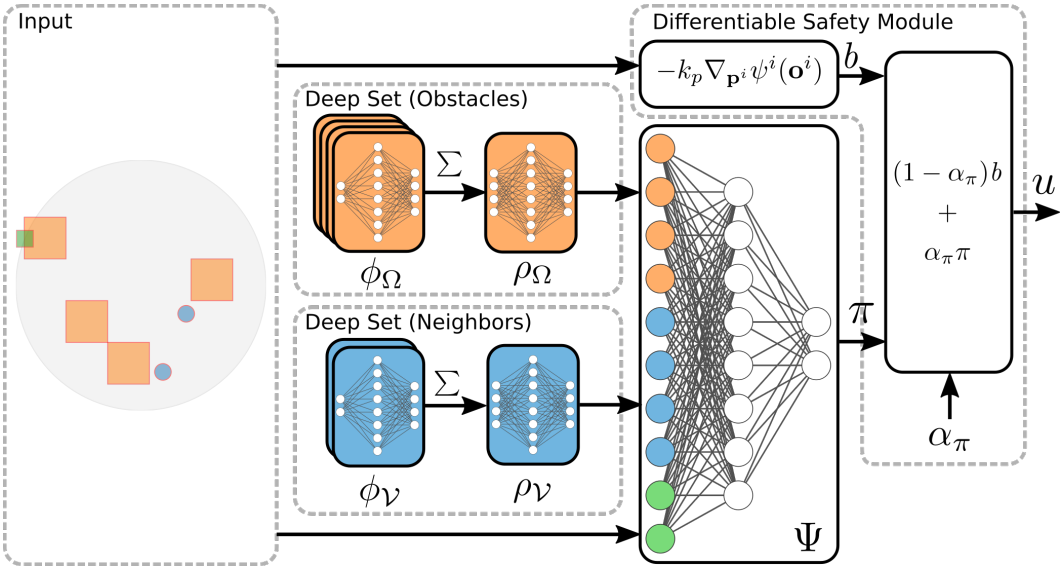


Figure 5.1: Neural network architecture consisting of 5 feed-forward components.

5.3 Method

In this section, we derive the method of **GLAS**, **G**lobal-to-**L**ocal **A**utonomy **S**ynthesis, to find policy u:

$$u(o^i) = \alpha(o^i)\pi(o^i) + (1 - \alpha(o^i))b(o^i), \tag{5.6}$$

where $\pi : O \to U$ is a learned function, $b : O \to U$ is a safety control module to ensure safety, and $\alpha : O \to [0, 1]$ is an adaptive gain function. We discuss each of the components of the controller in this section. The overview of our controller architecture is shown in Fig. 5.1.

Neural Policy Synthesis via Deep Imitation Learning

We describe how to synthesize the neural policy π that imitates the behavior of an expert, where the expert refers to a global optimal planner. Explicitly, we take batches of observation-action pairs from an expert demonstration dataset and we train a neural network by minimizing the loss on output action given an observation. To use this method, we need to generate an observation-action pair dataset from expert demonstration and design a deep learning architecture compatible with dynamic sensing network topologies.

Generating Demonstration Data

Our dataset is generated using expert demonstrations from an existing centralized planner [2]. This planner is resolution-complete and avoids local minima; it is com-

putationally efficient so we can generate large expert demonstration datasets; and it uses an optimization framework that can minimize control effort, so the policy imitates a solution with high performance according to the previously defined metrics. Specifically, we create our dataset by generating maps with (i) fixed-size static obstacles with random uniformly sampled grid positions and (ii) start/goal positions for a variable number of robots, and then by computing trajectories using the centralized planner. For each timestep and robot, we retrieve the local observation, o^i by masking the non-local information with the observation model h, and retrieving the action, u^i through the appropriate derivative of the robot i trajectory. We repeat this process n_{case} times for each robot/obstacle case. Our dataset, $\mathcal{D}$, is:

$$\mathcal{D} = \{(o^i, u^i)_k \mid \forall i \in \mathcal{I}, \forall k \in \{1 \dots n_{\text{case}}\}, \forall t\}. \tag{5.7}$$

Model Architecture with Deep Sets

The number of visible neighboring robots and obstacles can vary dramatically during each operation, which causes the dimensionality of the observation vector to be time-varying. Leveraging the permutation invariance of the observation, we model variable number of robots and obstacles with the Deep Set architecture [3, 4]. Theorem 7 from [3] establishes this property:

Theorem 6 *Let $F : [0, 1]^l \to \mathbb{R}$ be a permutation invariant continuous function iff it has the representation:*

$$f(x_1, \dots, x_l) = \rho\left(\sum_{m=1}^{l} \phi(x_m)\right), \tag{5.8}$$

for some continuous outer and inner function $\rho : \mathbb{R}^{l+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^{l+1}$, respectively. The inner function ϕ is independent of the function F.

Intuitively, the ϕ function acts as a contribution from each element in the set, and the ρ function acts to combine the contributions of each element. In effect, the policy can learn the contribution of the neighboring set of robots and obstacles with the following network structure:

$$\begin{aligned} \pi(o^i)_{\text{n}} &= \Psi([\rho_\Omega(\sum_{j \in \mathcal{N}^i_\Omega} \phi_\Omega(x^{ij})); \rho_\mathcal{I}(\sum_{j \in \mathcal{N}^i_V} \phi_\mathcal{I}(x^{ij}))]), \\ \pi(o^i) &= \pi(o^i)_{\text{n}} \ \min\{\frac{\pi_{\max}}{||\pi(o^i)_{\text{n}}||_2}, 1\}, \end{aligned} \tag{5.9}$$

where the semicolon denotes a stacked vector and $\Psi, \rho_\Omega, \phi_\Omega, \rho_\mathcal{I}, \phi_\mathcal{I}$ are feed-forward networks of the form:

$$\text{FF}(x) = W^l \sigma(\ldots W^1 \sigma(x)), \tag{5.10}$$

where FF is a feed-forward network on input x, W^l is the weight matrix of the lth layer, and σ is the activation function. We define the parameters for each of the 5 networks in Sec. 5.5. We also scale the output of the π network to always be less than $\pi_{\max}$, to maintain consistency with our baselines.

End-to-End Training

We train the neural policy π with knowledge of the safety module b, to synthesize a controller u with symbiotic components. We train through the output of u, not π, even though b has no tunable parameters. In effect, the parameters of π are updated such that the policy π smoothly interacts with b while imitating the global planner. With respect to a solution that trains through the output of π, end-to-end learning generates solutions with lower control effort, measured through the r_p metric (5.21), see Fig. 5.2.

Additional Methods in Training

We apply additional preprocessing methods to the observation to improve our training process performance and to regularize the data. We denote the difference between the original observation and the preprocessed data with an apostrophe; e.g., the input to the neural network is an observation vector denoted by $o^{i\prime}$.

We scale the relative goal vector observation as follows:

$$e^{ii\prime} = \alpha_g e^{ii}, \text{ where } \alpha_g = \min\{\frac{r_{\text{sense}}}{||e^{ii}||}, 1\}. \tag{5.11}$$

This regularizes cases when the goal is beyond the sensing radius. In such cases, the robot needs to avoid any robots/obstacles and continue toward the goal. However, the magnitude of e^{ii} outside the sensing region is not important.

We cap the maximum cardinality of the neighbor and obstacle sets with $\overline{\mathcal{N}_\mathcal{I}}$ and $\overline{\mathcal{N}_\Omega}$, e.g.,

$$\mathcal{N}_\mathcal{I}^{i\prime} = \{j \in \mathcal{N}_\mathcal{I}^i \mid \overline{\mathcal{N}_\mathcal{I}}\text{-closest robots w.r.t. } \|p^{ij}\|\}. \tag{5.12}$$

This enables batching of observation vectors into fixed-dimension tensors for fast training, and upper bounds the evaluation time of π to guarantee real-time performance on hardware in large swarm experiments.

5.4 Theoretical Analysis

We adopt the formulation of safe sets used in control barrier functions and define the global safe set X_s as the super-level set of a global safety function $g : X \to \mathbb{R}$. We define this global safety as the minimum of local safety functions, $h : O \to \mathbb{R}$ that specify pairwise collision avoidance between all objects in the environment:

$$X_s = \{x \in X \mid g(x) > 0\}, \tag{5.13}$$

$$g(x) = \min_{i,j} h(\overline{p}^{ij}), \quad h(\overline{p}^{ij}) = \frac{\|\overline{p}^{ij}\| - r_{\text{safe}}}{r_{\text{sense}} - r_{\text{safe}}}, \tag{5.14}$$

where $\overline{p}^{ij}$ denotes the vector between the closest point on object j to center of object i. This allows us to consistently define r_{safe} as the radius of the robot, where $r_{\text{safe}} < r_{\text{sense}}$. Intuitively, if a collision occurs between robots i, j, then $h(\overline{p}^{ij}) < 0$ and $g(x) < 0$, implying that the system is not safe. In order to synthesize local controls with guaranteed global safety, we need to show non-local safety functions cannot violate global safety. Consider a pair of robots outside of the neighborhood, $\|\overline{p}^{ij}\| > r_{\text{sense}}$. Clearly, $h(\overline{p}^{ij}) > 1$, implying this interaction is always safe.

Controller Synthesis

We use these safety functions to construct a global potential function, $\Psi : X \to \mathbb{R}$ that becomes unbounded when any safety is violated, which resembles logarithmic barrier functions used in the interior point method in optimization [5]. Similarly, we can construct a local function, $\Psi^i : O \to \mathbb{R}$:

$$\Psi(x) = -\log \prod_i \prod_{j \in \mathcal{N}^i} h(\overline{p}^{ij}), \tag{5.15}$$

$$\Psi^i(o^i) = -\log \prod_{j \in \mathcal{N}^i} h(\overline{p}^{ij}). \tag{5.16}$$

We use the local potential Ψ^i to synthesize the safety control module b. We first state some assumptions.

Assumption 2 *Initially, the distance between all objects is at least $r_{\text{safe}} + \Delta_r$, where Δ_r is a user-specified parameter.*

Assumption 3 *We assume robot i's geometry not to exceed a ball of radius r_{safe} centered at p^i.*

Here we present the single integrator case:

Theorem 7 *For the single integrator dynamics* (5.3)*, the safety defined by* (5.13) *is guaranteed under the control law* (5.6) *with the following* $b(o^i)$ *and* $\pi(o^i)$ *for a scalar gains* $k_p > 0$ *and* $k_c > 0$*:*

$$b(o^i) = -k_p \nabla_{p^i} \Psi^i(o^i) \tag{5.17}$$

$$\alpha_\pi(o^i) = \begin{cases} \frac{(k_p - k_c)\|\nabla_{p^i}\Psi^i(o^i)\|^2}{k_p\|\nabla_{p^i}\Psi^i(o^i)\|^2 + |\langle \nabla_{p^i}\Psi^i(o^i), \pi(o^i)\rangle|} & \Delta_h(o^i) < 0 \\ 1 & else \end{cases} \tag{5.18}$$

with $\Delta_h(o^i) = \min_{j \in \mathcal{N}^i} h(\overline{p}^{ij}) - \Delta_r$*, and* $\nabla_{p^i}\Psi^i$ *as in* (A.97)*.*

Next, we present the double integrator case:

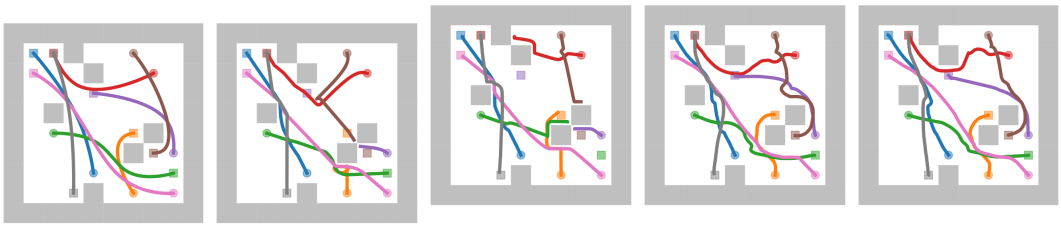
Theorem 8 *For the double integrator dynamics given in* (5.3)*, the safety defined by* (5.13) *is guaranteed under control law* (5.6) *for the barrier-controller and the gain defined as:*

$$\begin{aligned} b &= -k_v(v^i + k_p \nabla_{p^i}\Psi^i) - k_p \frac{d}{dt}\nabla_{p^i}\Psi^i - k_p \nabla_{p^i}\Psi^i, \\ \alpha_\pi &= \begin{cases} \frac{a_1 - k_c(k_p\Psi^i + \frac{1}{2}\|v-k\|^2)}{a_1 + |a_2|} & \Delta_h(o^i) < 0 \\ 1 & else \end{cases}, \\ a_1 &= k_v\|v^i + k_p \nabla_{p^i}\Psi^i\|^2 + k_p^2\|\nabla_{p^i}\Psi^i\|^2, \\ a_2 &= \langle v^i, k_p \nabla_{p^i}\Psi^i\rangle + \langle v^i + k_p\nabla_{p^i}\Psi^i, \pi + k_p \frac{d}{dt}\nabla_{p^i}\Psi^i\rangle, \end{aligned} \tag{5.19}$$

where $k_p > 0$*,* $k_c > 0$*, and* $k_v > 0$ *are scalar gains,* Δ_h *is defined as in Theorem 7,* $\frac{d}{dt}\nabla_{p^i}\Psi^i$ *is defined in* (A.105) *and the dependency on the observation is suppressed for legibility.*

We make some remarks on the results of the proofs.

Remark 3 *The setup for this proof is to give* π *maximal authority without violating safety. This trade-off is characterized through the design parameter* Δ_r*, and the gains* k_p *and* k_v *that control the measure of the conservativeness of the algorithm. For the discrete implementation of this algorithm, we introduce a parameter,* $\varepsilon \ll 1$*, to artificially decrease* α_π *such that* $\alpha_\pi = 1 - \varepsilon$ *when* $\Delta_h > 0$*. The results (in continuous time) of the proof still hold as* α_π *is a scalar gain on a destabilizing term, so a lower* α_π *further stabilizes the system. The gain* k_c *can be arbitrarily small, in simulation we set it to* 0*.*



(a) Global (b) ORCA (c) GLAS Barrier (d) GLAS Two-stage (e) GLAS End-to-end

Figure 5.2: Example trajectories for baselines (a-c) and GLAS method (d,e).

Remark 4 *Intuitively, $\Delta_h(o^i)$ defines an unsafe domain for robot i. In safe settings, i.e. $\Delta_h > 0$, $\alpha_\pi = 1 - \varepsilon$ and so the barrier has little effect on the behavior. In most unsafe cases, the barrier will be activated, driving a large magnitude safety response. However, in dense multi-robot settings, it is possible for safety responses to cancel each other out in a gridlock, resulting in dangerous scenarios where small disturbances can cause the system to violate safety. In this case, we use the above result to put an adaptive gain, π (5.18,5.19) on the neural policy and to drive α_π to 0, cancelling the effect of α_π. Thus, we use a convex combination of the neural optimal policy and the safety control module to guarantee safety in all cases.*

Remark 5 *In Theorem 7, we synthesize a local nominal control that guarantees the global safety of the system. In Theorem 8, we use Lyapunov backstepping to provide the same nominal control through a layer of dynamics. This method is valid for a large class of nonlinear dynamical systems known as full-state feedback linearizeable systems [6]. By following this method, GLAS can be extended to other nonlinear dynamical systems in a straightforward manner.*

5.5 Simulation Results

Performance Metrics:

To evaluate performance, we have two criteria as specified by the optimal control problem. We define our metrics over the set of successful robots, $\mathcal{I}_s$, that reach their goal and have no collisions:

$$\mathcal{I}_s = \{i \in \mathcal{I} \mid x^i(t_f) = g^i \text{ and } \|p_t^{ij}\| > r_{\text{safe}},\ \forall j, t\}. \tag{5.20}$$

Our first metric of success, r_s, is the number of successful robots, and our second metric, r_p, is the cost of deploying a successful robot trajectory. For example, if the

cost function $c(x, u)$ is the total control effort, the performance metrics are:

$$r_s = |\mathcal{I}_s|, \quad \text{and} \quad r_p = \sum_{i \in \mathcal{I}} \int_0^{t_f} \|u^i\|_2 dt. \tag{5.21}$$

We now present results of simulation comparing GLAS and its variants with state-of-the-art baselines as well as experimental results on physical quadrotors. Our supplemental video includes additional simulations and experiments.

Learning Implementation and Hyperparameters

For data generation, we use an existing implementation of a centralized global trajectory planner [2] and generate $\approx 2{\times}10^5$ (200 k) demonstrations in random 8 m $\times$ 8 m environments with 10 or 20 % obstacles randomly placed in a grid pattern and 4, 8, or 16 robots (e.g., see Fig. 5.2 for 10 % obstacles and 8 robots). We sample trajectories every 0.5 s and generate $|\mathcal{D}| = 40{\times}10^6$ (40 M) data points in total, evenly distributed over the 6 different environment kinds. We use different datasets for single and double integrator dynamics with different desired smoothness in the global planner.

We implement our learning framework in Python using PyTorch [7]. The ϕ_Ω and ϕ_V networks have an input layer with 2 neurons, one hidden layer with 64 neurons, and an output layer with 16 neurons. The ρ_Ω and ρ_V networks have 16 neurons in their input and output layers and one hidden layer with 64 neurons. The Ψ network has an input layer with 34 neurons, one hidden layer with 64 neurons, and outputs π using two neurons. All networks use a fully connected feedforward structure with ReLU activation functions. We use an initial learning rate of 0.001 with the PyTorch optimizer ReduceLROnPlateau function, a batch size of 32 k, and train for 200 epochs. During manual, iterative hyperparameter tuning, we found that the hidden layers should at least use 32 neurons. For efficient training of the Deep Set architecture, we create batches where the number of neighbors $|\mathcal{N}_V|$ and number of obstacles $|\mathcal{N}_\Omega|$ are the same and limit the observation to a maximum of 6 neighbors and 6 obstacles.

GLAS Variants

We study the effect of each component of the system architecture by comparing variants of our controller: *end-to-end*, *two-stage*, and *barrier*. End-to-end and two-stage are synthesized through (5.6), but differ in how π is trained. For end-to-end we calculate the loss on $u(o^i)$, while for two-stage we calculate the loss on

$\pi(o^i)$. Comparing these two methods isolates the effect of the end-to-end training. The barrier variant is a linear feedback to goal controller with our safety module. Essentially, barrier is synthesized with (5.6), where the π heuristic is replaced with a linear goal term: Ke^{ii}, where, for single integrator systems, $K = k_p I$, and for double integrator systems, $K = [k_p I, k_v I]$, with scalar gains k_p and k_v. Studying the performance of the barrier variant isolates the effect of the global-to-local heuristic training.

Single Integrator Dynamics

We compare our method with ORCA, a state-of-the-art decentralized approach for single integrator dynamics. Unlike GLAS, ORCA requires relative velocities with respect to neighbors in addition to relative positions. All methods compute a velocity action with guaranteed safety.

We show example trajectories for the global planner, ORCA, and GLAS variants in Fig. 5.2. In Fig. 5.2b/5.2c, the purple and brown robots are getting stuck in local minima caused by obstacle traps. In Fig. 5.2d/5.2e, our learned policies are able to avoid those local minima. The end-to-end approach produces smoother trajectories that use less control effort, e.g., red and brown robot trajectories in Fig. 5.2e.

Evaluation of Metrics

We deploy the baseline and variants over 100 validation cases with 2, 4, 8, 16, and 32 robots and 10 % and 20 % obstacle density (10 validation instances for each case) and empirically evaluate the metrics defined in (5.21). Our training data only contains different examples with up to 16 robots. We train 5 instances of end-to-end and two-stage models, to quantify the effect of random-weight initialization in the neural networks on performance, see Fig. 5.3.

In the top row, we consider the success metric r_s. In a wide range of robot/obstacle cases (2–16 robots/64 m^2), our global-to-local methods outperform ORCA by 20 %, solving almost all instances. Our barrier variant has a similar success rate as ORCA, demonstrating that the neural heuristic π is crucial for our high success rates. The two-stage approach generalizes better to higher-density cases beyond those in the training data. We observe the inverse trend in the double integrator case and analyzing this effect is an interesting future direction.

In the bottom row, we measure control effort r_p. Our end-to-end approach uses less control effort than the two-stage approach. ORCA has the lowest control effort,

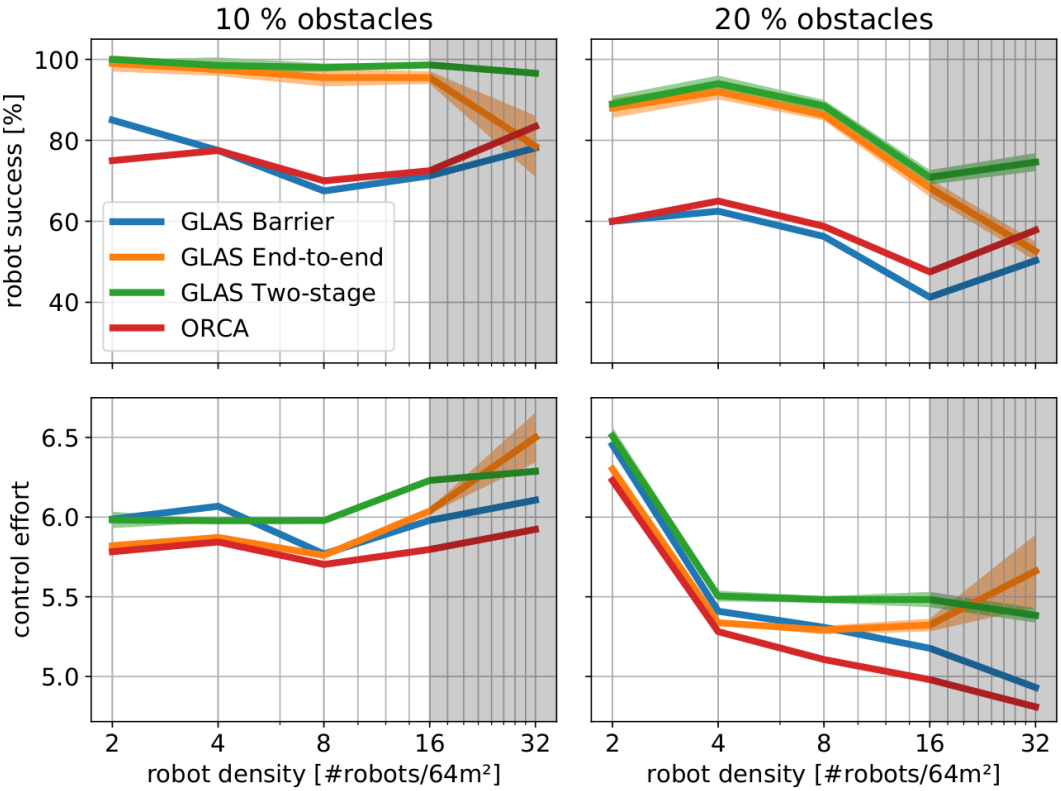


Figure 5.3: Success rate and control effort with varying numbers of robots in a $8\,\mathrm{m} \times 8\,\mathrm{m}$ space for single integrator systems. Shaded area around the lines denotes standard deviation over 5 repetitions. The shaded gray box highlights validation outside the training domain.

because the analytical solution to single integrator optimal control is a bang-bang controller, similar in nature to ORCA's implementation.

Effect of Complexity of Data on Loss Function

At some robot/obstacle density, local observations and their actions will become inconsistent, i.e., the same observation will match to different actions generated by the global planner. We quantify this data complexity limit by recording the value of the validation loss when training using datasets of varying complexity, see Fig. 5.4.

Here, we train the end-to-end model with $|\mathcal{D}| = 5\,\mathrm{M}$ using isolated datasets of 10 and 20 % obstacles and with 4 and 16 robots, as well as a mixed dataset. We see that the easiest case, 4 robots and 10 % obstacles, results in a the smallest loss, roughly 1 % of the maximum action magnitude of the expert. The learning task is more difficult with high robot density compared to high obstacle density. A mixed dataset, as used in all other experiments, is a good trade-off between imitating the expert very well and being exposed to complex situations.

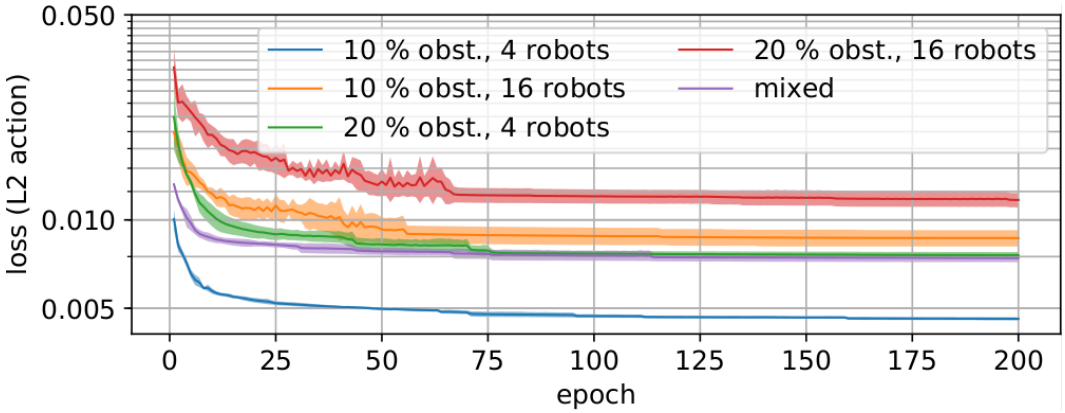


Figure 5.4: Testing loss when training using 10 and 20 % obstacles and 4 or 16 robots. Synthesizing a distributed policy that is consistent with the global data is harder for high robot densities than for high obstacle densities. We use the GLAS end-to-end with $|\mathcal{D}| = 5$ M and repeat 5 times.

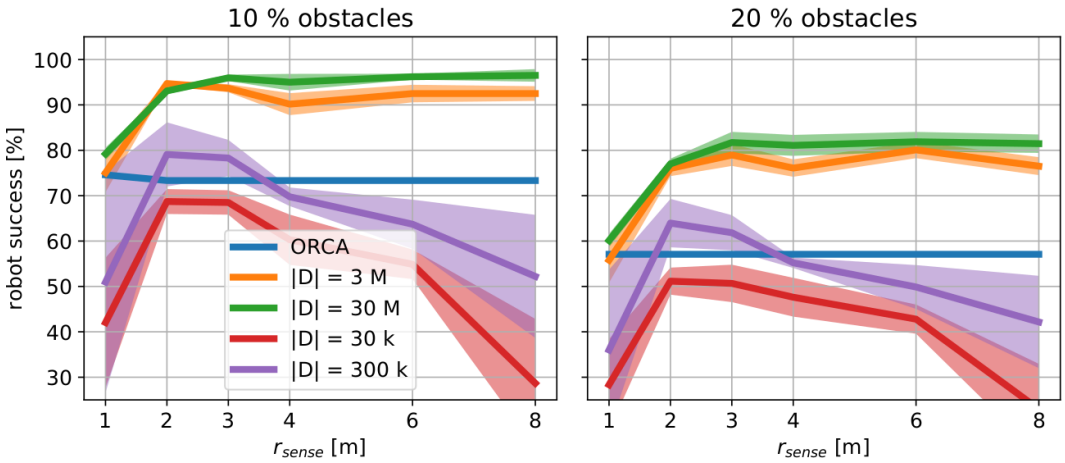


Figure 5.5: Effect of sensing radius and amount of training data on robot success rate. The validation has 4, 8, and 16 robot cases with 10 instances each. Training and validation were repeated 5 times; the shaded area denotes the standard deviation.

Effect of Radius of Sensing on Performance

We quantify the transition from local-to-global by evaluating the performance of models trained with various sensing radii and dataset size. We evaluate performance on a validation set of 4, 8, and 16 robot cases with 10 % and 20 % obstacle densities. First, we found that there exists an optimal sensing radius for a given amount of data, which increases with larger datasets. For example, in the 20 % obstacle case, the optimal sensing radius for $|\mathcal{D}| = 300$ k is around 2 m and the optimal radius for $|\mathcal{D}| = 30$ M is 8 m. Second, we found that between models of various dataset sizes

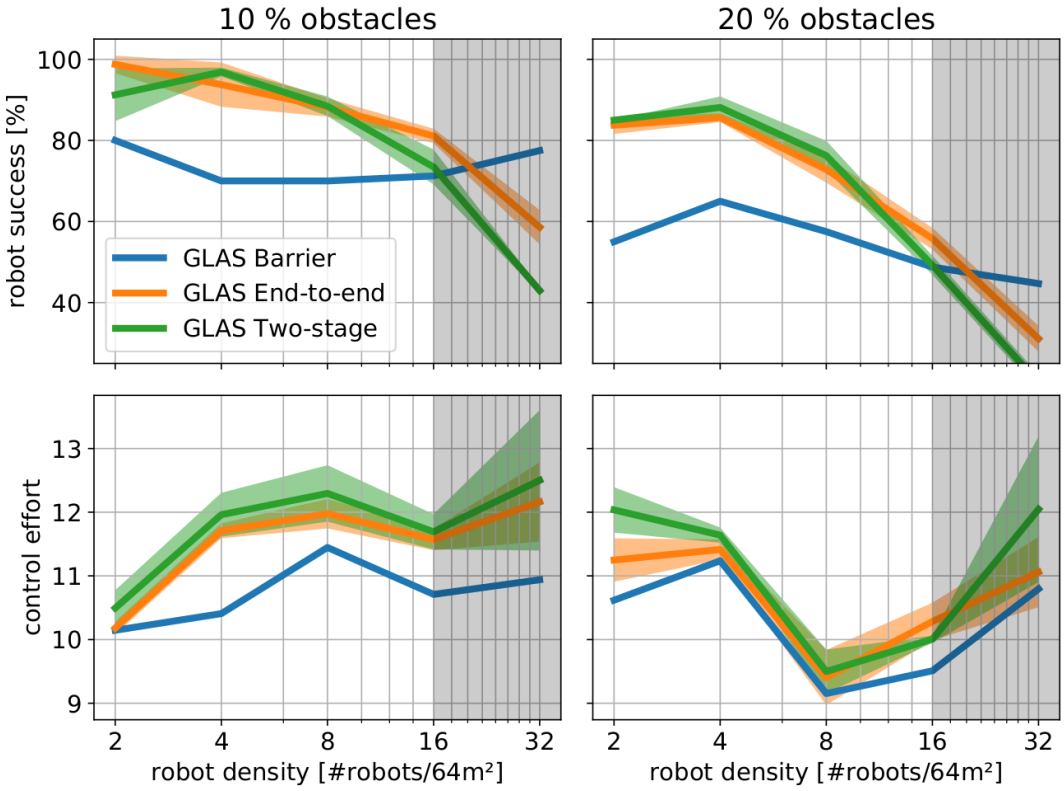


Figure 5.6: Success rate and control effort with varying numbers of robots in a $8\,\mathrm{m} \times 8\,\mathrm{m}$ workspace for double integrator systems. Shaded area around the lines denotes standard deviation over 5 repetitions. The shaded gray box highlights validation outside the training domain.

the performance gap at small sensing radii is smaller compared to the performance gap at large sensing radii. This result suggests that little data is needed to use local information well, and large amounts of data is needed to learn from global data.

Double Integrator Dynamics

We extend our results to double integrator dynamical systems to demonstrate GLAS extends naturally as a dynamically-coupled motion planner. Similar to the single integrator evaluation, we show double integrator statistical evaluation with respect to the performance metrics (5.21), for varying robot and obstacle density cases. We use the same setup as in the single integrator case (see Sec. 5.5), but with a different dataset ($|\mathcal{D}| = 20\,\mathrm{M}$). Results are shown in Fig. 5.6.

In the top row, we consider the success metric r_s. In a wide range of robot density cases (2–16 robots/$64\,\mathrm{m}^2$), our global-to-local end-to-end method again outperforms the barrier baseline by 15 % to 20 %. In the 32 robots/$64\,\mathrm{m}^2$ case, the barrier baseline outperforms the global-to-local methods. We suspect this is a com-

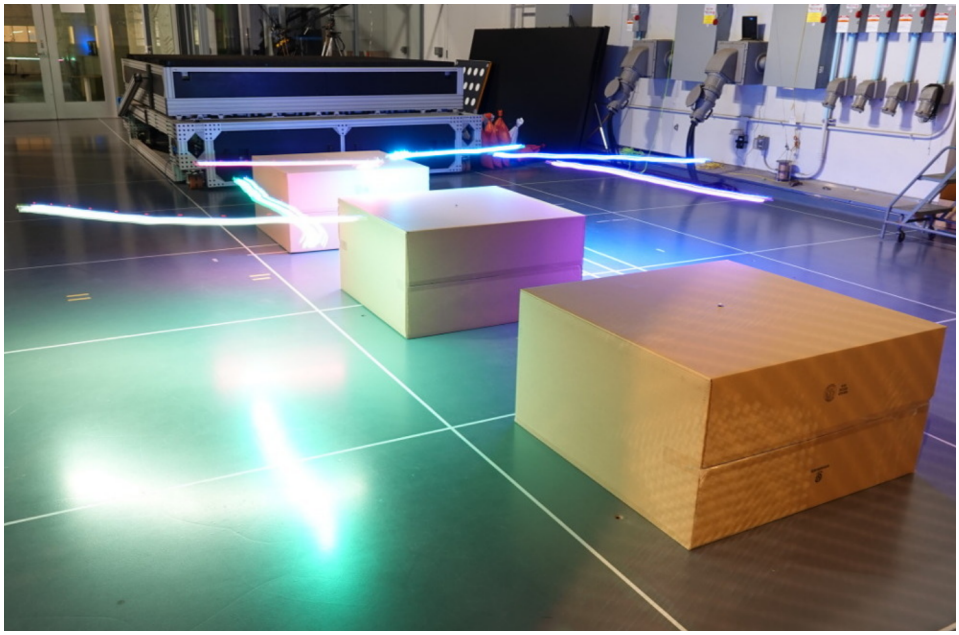


Figure 5.7: GLAS Hardware Experiment

bination of our method suffering from the significantly higher complexity of the problem and the naive barrier method performing well because the disturbances from the robot interaction push the robot out of local minima obstacle traps. In contrast to the single integrator case, the end-to-end solution generalizes better than the two-stage in higher robot densities. We conjecture that having a much larger training set can significantly improve performance.

In the bottom row, we consider the performance metric r_p. For double integrator systems, the cost function $c(x, u)$ corresponds to the energy consumption of each successfully deployed robot. The end-to-end variant uses less effort ($\leq$ 6.25 %) than the two-stage method on average.

5.6 Hardware Results

We implement the policy evaluation (π, α_π, b) in C to enable real-time execution on-board of Crazyflie 2.0 quadrotors using double integrator dynamics (see Fig. 5.7). The quadrotors use a small STM32 microcontroller with 192 kB SRAM running at 168 MHz. Our policy evaluation takes 3.4 ms for 1 neighbor and 5.0 ms for 3 neighbors, making it computationally efficient enough to execute our policy in real-time at 40 Hz. On-board, we evaluate the policy, forward-propagate double integrator dynamics, and track the resulting position and velocity setpoint using a nonlinear controller. The experimental validation demonstrates that our policy generalizes to

novel environments where the obstacles are arranged in continuous space, as opposed to on a grid.

We use a double integrator GLAS end-to-end policy in three different scenarios with up to 3 obstacles and 12 quadrotors. We fly in a motion capture space, where each robot is equipped with a single marker, using the Crazyswarm [8] for tracking and scripting. Our demonstration shows that our policy works well on robots and that it can also handle cases that are considered difficult in decentralized multi-robot motion planning, such as swapping positions with a narrow corridor.

5.7 Related Work

Multi-robot motion planning is an active area of research because it is a non-convex optimization problem with high state and action dimensionality. We compare the present work with state-of-the-art methods: (a) collision avoidance controllers, (b) optimal motion-planners, and (c) deep-learning methods.

Collision Avoidance: Traditional controller-level approaches include Optimal Reciprocal Collision Avoidance (ORCA) [9], Buffered Voronoi Cells [10], Artificial Potential Functions [11–13], and Control Barrier Functions [14]. These methods are susceptible to trapping robots in local minima. We address this problem explicitly by imitating a complete global planner with local information. For optimal performance, we propose to learn a controller end-to-end, including the safety module. Existing methods that are based on optimization [9, 10, 14] are challenging for backpropagation for end-to-end training. Existing analytic methods [11] do not explicitly consider gridlocks, where robots' respective barriers cancel each other and disturbances can cause the system to violate safety. Thus, we derive a novel differentiable safety module. This system design of fully differentiable modules for end-to-end backpropagation is also explored in reinforcement learning [15] and estimation [16].

Motion Planners: Motion planners are a higher-level approach that explicitly solves the optimal control problem over a time horizon. Solving the optimal control problem is non-convex, so most recent works with local guarantees use approximate methods like Sequential Convex Programming to quickly reach a solution [2, 17]. Motion planners are distinguished as either global and centralized [2] or local and decentralized [17, 18], depending on whether they find solutions in joint space or computed by each robot.

GLAS is inherently scalable because it is computed at each robot. Although it has

no completeness guarantee, we empirically show it avoids local minima more often than conventional local methods. The local minima issue shared by all the local methods is a natural trade-off of decentralized algorithms. Our method explores this trade-off explicitly by imitating a complete, global planner with only local information.

Deep Learning Methods: Recently, there have been new learning-based approaches for multi-robot path planning [19–23]. These works use deep Convolutional Neural Networks (CNN) in a discrete state/action domain. Such discretization prevents coupling to higher-order robot dynamics whereas our solution permits tight coupling to the system dynamics by operating in a continuous state/action domain using a novel network architecture based on Deep Sets [3]. Deep Sets are a relatively compact representation that leverages the permutation-invariant nature of the underlying interactions, resulting in a less computationally-expensive solution than CNNs. In contrast to our work, the Neural-Swarm approach [4] uses Deep Sets to augment a tracking controller for close proximity flight.

Imitation Learning (IL) can imitate an expensive planner [19–21, 24], thereby replacing the optimal control or planning solver with a function that approximates the solution. GLAS uses IL and additionally changes the input domain from full state information to a local observation, thereby enabling us to synthesize a decentralized policy from a global planner.

BIBLIOGRAPHY

[1] Benjamin Rivière, Wolfgang Hönig, Yisong Yue, and Soon-Jo Chung. "GLAS: Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning With End-to-End Learning". In: *IEEE Robotics Automation Letters* 5.3 (2020), pp. 4249–4256. DOI: 10.1109/LRA.2020.2994035.

[2] Wolfgang Hönig et al. "Trajectory Planning for Quadrotor Swarms". In: 34.4 (2018), pp. 856–869. URL: https://doi.org/10.1109/TRO.2018.2853613.

[3] Manzil Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 3391–3401. URL: http://papers.nips.cc/paper/6931-deep-sets.

[4] Guanya Shi et al. "Neural-Swarm: Decentralized Close-Proximity Multirotor Control Using Learned Interactions". In: *Proc. IEEE Int. Conf. Robot. Autom.* 2020. URL: https://arxiv.org/abs/2003.02992.

[5] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014. ISBN: 978-0-521-83378-3. DOI: 10.1017/CBO9780511804441. URL: https://web.stanford.edu/%5C%7Eboyd/cvxbook/.

[6] Hassan K Khalil. *Nonlinear systems*. Upper Saddle River, NJ: Prentice-Hall, 2002. ISBN: 978-0130673893. URL: https://cds.cern.ch/record/1173048.

[7] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[8] James A. Preiss* et al. "Crazyswarm: A large nano-quadcopter swarm". In: *Proc. IEEE Int. Conf. Robot. Autom.* 2017, pp. 3299–3304. URL: https://doi.org/10.1109/ICRA.2017.7989376.

[9] Jur van den Berg et al. "Reciprocal *n*-Body Collision Avoidance". In: *Int. Symp. on Robot. Res.* Vol. 70. Springer Tracts in Advanced Robotics. Springer, 2009, pp. 3–19. URL: https://doi.org/10.1007/978-3-642-19457-3%5C_1.

[10] Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y. Hadaegh. "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents". In: 33.5 (2017), pp. 1103–1123. DOI: 10.1109/TRO.2017.2705044. URL: https://doi.org/10.1109/TRO.2017.2705044.

[11] Oussama Khatib. “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots”. In: *Autonomous Robot Vehicles*. Springer, 1990, pp. 396–404. URL: https://doi.org/10.1007/978-1-4613-8997-2%5C_29.

[12] Elon Rimon and Daniel E. Koditschek. “Exact robot navigation using artificial potential functions”. In: 8.5 (1992), pp. 501–518. URL: https://doi.org/10.1109/70.163777.

[13] Herbert G. Tanner and Amit Kumar. “Formation Stabilization of Multiple Agents Using Decentralized Navigation Functions”. In: *Robotics: Science & Systems*. 2005, pp. 49–56. URL: http://www.roboticsproceedings.org/rss01/p07.html.

[14] Li Wang, Aaron D. Ames, and Magnus Egerstedt. “Safety Barrier Certificates for Collisions-Free Multirobot Systems”. In: 33.3 (2017), pp. 661–674. URL: https://doi.org/10.1109/TRO.2017.2659727.

[15] Richard Cheng et al. “Control Regularization for Reduced Variance Reinforcement Learning”. In: *Proc. Int. Conf. Machine Learning*. Vol. 97. 2019, pp. 1141–1150. URL: http://proceedings.mlr.press/v97/cheng19a.html.

[16] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. “Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors”. In: *Robotics: Science & Systems*. 2018. URL: http://www.roboticsproceedings.org/rss14/p01.html.

[17] Daniel Morgan et al. “Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming”. In: *I. J. Robotics Res.* 35.10 (2016), pp. 1261–1285. URL: https://doi.org/10.1177/0278364916632065.

[18] Carlos E. Luis and Angela P. Schoellig. “Trajectory Generation for Multiagent Point-To-Point Transitions via Distributed Model Predictive Control”. In: 4.2 (2019), pp. 375–382. URL: https://doi.org/10.1109/LRA.2018.2890572.

[19] Guillaume Sartoretti et al. “PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning”. In: 4.3 (2019), pp. 2378–2385. URL: https://doi.org/10.1109/LRA.2019.2903261.

[20] Qingbiao Li et al. “Graph Neural Networks for Decentralized Multi-Robot Path Planning”. In: *CoRR* abs/1912.06095 (2019). URL: http://arxiv.org/abs/1912.06095.

[21] Arbaaz Khan, Vijay Kumar, and Alejandro Ribeiro. “Graph Policy Gradients for Large Scale Unlabeled Motion Planning with Constraints”. In: *CoRR* abs/1909.10704 (2019). URL: http://arxiv.org/abs/1909.10704.

[22] Arbaaz Khan et al. “Learning Safe Unlabeled Multi-Robot Planning with Motion Constraints”. In: *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* 2019, pp. 7558–7565. DOI: 10.1109/IROS40897.2019.8968483. URL: https://doi.org/10.1109/IROS40897.2019.8968483.

[23] Dhananjay Raju, Suda Bharadwaj, and Ufuk Topcu. “Decentralized Runtime Synthesis of Shields for Multi-Agent Systems”. In: *CoRR* abs/1910.10380 (2019). URL: http://arxiv.org/abs/1910.10380.

[24] Yunpeng Pan et al. “Agile Autonomous Driving using End-to-End Deep Imitation Learning”. In: *Robotics: Science & Systems*. 2018. URL: http://www.roboticsproceedings.org/rss14/p56.html.

Chapter 6

CONCLUSION

6.1 Contributions

My research exists at the intersection of optimal control and reinforcement learning and develops algorithms that compute trajectories in real-time and converge to the globally-optimal solution. In Chapter 2, we present a global solution to the classic optimal control problem with complex dynamics. In Chapter 3, we present a global solution to the partially-observeable optimal control problem provided with a filter, which can be derived for the case of sensor and actuator failures. In Chapter 4, we present an algorithmic solution to N-player games using data-driven, decentralized heuristics, supported by global-to-local learning techniques developed in Chapter 5.

The ability to solve a spectrum of important problem settings with a common algorithm blurs the definition of Monte Carlo Tree Search between numerical algorithm and a more general intelligence: if a robot can predict its effect on the future, use that information to refine its plan, accumulate memory with neural heuristics, and perform these processes efficiently enough, it will have sufficient autonomy to explore caves on the moon, drive a car, or compete in sports. In addition, the robot's thought process will be interpretable by humans through analysis of the tree data structure. For example, we will be able to ask: what was the robot's understanding of the world state? what future did the robot believe its actions were leading to? and why did it think this behavior was optimal? This thesis build towards this vision, provide fundamental results, and suggests exciting new research directions.

6.2 Future Work

Revisiting Foundations of Global Convergence

In Chapters 2 and 3, we manipulated representation of continuous space to apply the UCT global optimality for discrete systems. As discussed in Chapter 1, the UCT result uses results from the online learning field to balance exploration and exploitation: nonstationary concentration inequalities are used to estimate the value of the tree policy and while simultaneously changing the tree policy. Although empirically successful, this strategy is not satisfying theoretically because it hides the planning process evolution in the non-stationarity assumptions. An alternative approach could be to fix the tree policy, consider the visit count of nodes in the tree

as a Markov Chain, and analyze the conditions of its convergence as the number of visits in the tree grows. This alternative formulation would expose the dynamics of tree search and provide a more fulfilling understanding of real-time intelligence. Furthermore, an alternative theoretical foundation could provide a path forward for tighter instance-specific rates or convergence in sequential games.

Efficient Discrete Representation of Continuous Systems

In Chapter 2, numerical results demonstrated that a naive algorithm with an efficient representation will outperform a sophisticated algorithm in a poorly chosen representation. In addition, decision-making textbooks consider the problem data (state, action, dynamics, reward) a given, but that is not the case for a robot operating without limited human intervention. Therefore, there is a need for to develop rigorous approaches to construct efficient representations from sensor data. The field of machine learning studies variations of this problem, yet the field of robotics has an additional constraint that the resulting representation be interpretable by human users and designers.

Integrating Reasoning and Memory (Search and Deep Learning)

In Chapter 4, the Neural Tree Expansion (NTE) algorithm used data-driven heuristics and MCTS in a two-process model that smoothly transitions between computationally-cheap, data-driven memorization via neural networks and computationally-expensive, causal reasoning with tree search. Although this combination has proven empirically successful in this work and others, the interface of deep learning and tree search is still ad-hoc because the effect of the heuristic on convergence is not formalized. Building a fundamental understanding of this memory/reasoning interface will allow us to continue aggressively pushing the limits of data-driven deep learning while inheriting real-time planning's robustness to domain shift, and search trace's explainability and interpretability.

Parallel Algorithm Design

In all the work we have presented, the robot's performance is dependent on the number of simulations it can generate in real-time. Depending on the problem's complexity, processor capability, and re-planning rate, the typical number of simulations in our experiments is between 100 to 10,000. However, all of these cases are limited because the UCT algorithm is not efficiently parallelized, and therefore the number of simulations are generated in a sequential, serial fashion. In fact, the

tree policy computation, which requires global memory based on aggregates of trajectory data, is very cheap compared to simulating trajectories, which requires only local memory. Therefore, if we imagine an algorithm that simulates the trajectories in parallel while not violating the convergence result, we can apply Amdahl's reasoning [1] to predict the algorithm will produce a large multiplicative factor more number of simulations in a fixed time. Applying the convergence result, this corresponds to more optimal real-time robot behavior.

BIBLIOGRAPHY

[1] Gene M Amdahl. “Validity of the single processor approach to achieving large scale computing capabilities”. In: *AFIPS Conference Proceedings* 30 (1967), pp. 483–485.

Appendix A

PROOFS

A.1 MCTS for Dynamics

In this section, unless otherwise specified, all norms are the 2-norm.

Proof of Lemma 2

Before we begin, we will introduce a piece of machinery to help us in the proof. This is a variation of the Discrete Gronwall Lemma [1] with a vanishing disturbance.

Lemma 4 *Let $(X, \|\cdot\|_X)$ be a metric space. Consider the sequence $\{a_n\}_{n\in\mathbb{N}} \subseteq X$ and a nonnegative real sequence $\{b_n\}_{n\in\mathbb{N}}$ satisfying*

$$\lim_{n\to\infty} a_n = a_\infty, \quad b_{n+1} \leq \gamma b_n + d(a_n) \tag{A.1}$$

where $d : X \to \mathbb{R}_{\geq 0}$. For $0 \leq \gamma < 1$, if d is bounded on X and continuous at a_∞ then

$$\limsup_{n\to\infty} b_n \leq \frac{1}{1-\gamma} d(a_\infty) \tag{A.2}$$

Proof 1 *Fix $\varepsilon > 0$. Denote the uniform bound of d as M: $|d(a)| \leq M \ \forall a \in X$.*

Let $\nu, \eta > 0$ be such that $\nu b_0 + \frac{\nu M}{1-\gamma} + \frac{\eta}{1-\gamma} < \varepsilon$.

Let N_1 be such that $|d(a_i) - d(a_\infty)| < \eta \ \forall i \geq N_1$. N_1 exists due to the convergence of $\{a_n\}_{n\in\mathbb{N}}$ and the continuity of d at a_∞.

Let N_2 be such that $\gamma^i < \nu \ \forall i \geq N_2$. For all $n \geq N_1 + N_2$, one can show by induction that:

$$b_n \leq \gamma^n b_0 + \sum_{i=0}^{n-1} \gamma^{n-1-i} d(a_i) \tag{A.3}$$

$$= \gamma^n b_0 + \sum_{i=0}^{N_1-1} \gamma^{n-1-i} d(a_i) + \sum_{i=N_1}^{n-1} \gamma^{n-1-i} d(a_i) \tag{A.4}$$

For the first term, we can see that as $n \geq N_2$, $\gamma^n b_0 \leq \nu b_0$.

In the second term, $n-1-i \geq N_2 \; \forall i = 0, \ldots N_1 - 1$. Therefore

$$\sum_{i=0}^{N_1-1} \gamma^{n-1-i} d(a_i) \leq \gamma^{N_2} \sum_{i=0}^{N_1-1} \gamma^{n-N_2-1-i} d(a_i) \leq \nu \sum_{i=0}^{N_1-1} \gamma^{n-N_2-1-i} d(a_i) \tag{A.5}$$

$$\leq \nu \sum_{i=0}^{N_1-1} \gamma^{n-N_2-1-i} M = \nu M \frac{\gamma^{n-N_1-N_2} - \gamma^{n-N_2}}{1-\gamma} \tag{A.6}$$

$$\leq \frac{\nu M}{1-\gamma} \tag{A.7}$$

For the third term, as $i \geq N_1$:

$$\sum_{i=N_1}^{n-1} \gamma^{n-1-i} d(a_i) = \sum_{i=N_1}^{n-1} \gamma^{n-1-i} (d(a_\infty) - d(a_i)) + \sum_{i=N_1}^{n-1} \gamma^{n-1-i} d(a_\infty) \tag{A.8}$$

$$\leq \sum_{i=N_1}^{n-1} \gamma^{n-1-i} |d(a_\infty) - d(a_i)| + \sum_{i=N_1}^{n-1} \gamma^{n-1-i} d(a_\infty) \tag{A.9}$$

$$\leq \sum_{i=N_1}^{n-1} \gamma^{n-1-i} \eta + \sum_{i=N_1}^{n-1} \gamma^{n-1-i} d(a_\infty) \tag{A.10}$$

$$= \eta \frac{1-\gamma^{n-N_1}}{1-\gamma} + d(a_\infty) \frac{1-\gamma^{n-N_1}}{1-\gamma} \tag{A.11}$$

$$\leq \frac{\eta}{1-\gamma} + \frac{d(a_\infty)}{1-\gamma} \tag{A.12}$$

Therefore

$$b_n \leq \nu b_0 + \frac{\nu M}{1-\gamma} + \frac{\eta}{1-\gamma} + \frac{d(a_\infty)}{1-\gamma} \leq \varepsilon + \frac{d(a_\infty)}{1-\gamma} \tag{A.13}$$

and observing that the bound is not a function of n,

$$\sup_{m \geq n} b_m \leq \varepsilon + \frac{d(a_\infty)}{1-\gamma} \tag{A.14}$$

Therefore, by the definition of a limit,

$$\limsup_{n \to \infty} b_n := \lim_{n \to \infty} \sup_{m \geq n} b_m \leq \frac{d(a_\infty)}{1-\gamma} \tag{A.15}$$

Now we will present the proof to Lemma 2.

Proof 2 *Consider two sequences of value functions formed through the Bellman iteration associated with each problem:*

$$V_{j+1} = (\mathcal{T}_1 V_j)(x) := \max_{u_1 \in U_1} R(F(x, u_1), u_1) + \gamma V_j(F(x, u_1)) \tag{A.16}$$

$$W_{j+1} = (\mathcal{T}_2 W_j)(x) := \max_{u_2 \in U_2} R(F(x, u_2), u_2) + \gamma W_j(F(x, u_2)) \tag{A.17}$$

For discounted MDPs ($\gamma < 1$), there exists an optimal value function, the unique fixed point of Bellman iteration. We show the limiting difference of the two sequences is bounded. Consider the sequence formed by the max norm of the difference between V_j and W_j:

$$Z_j := \|V_j - W_j\|_\infty = \max_{x \in X} |V_j(x) - W_j(x)| \tag{A.18}$$

Note that

$$\begin{aligned} Z_{j+1} &= \|V_{j+1} - W_{j+1}\|_\infty && \text{(A.19)} \\ &= \|\mathcal{T}_1 V_j - \mathcal{T}_2 W_j\|_\infty && \text{(A.20)} \\ &= \|\mathcal{T}_1 V_j - \mathcal{T}_1 W_j + \mathcal{T}_1 W_j - \mathcal{T}_2 W_j\|_\infty && \text{(A.21)} \\ &\leq \|\mathcal{T}_1 V_j - \mathcal{T}_1 W_j\|_\infty + \|\mathcal{T}_1 W_j - \mathcal{T}_2 W_j\|_\infty && \text{(A.22)} \end{aligned}$$

The first term is bounded through the contracting properties of the Bellman operator:

$$\|\mathcal{T}_1 V_j - \mathcal{T}_1 W_j\|_\infty \leq \gamma \|V_j - W_j\|_\infty = \gamma Z_j \tag{A.23}$$

Considering the second term, $\forall x \in X$:

$$\begin{aligned} &|\mathcal{T}_1 W_j(x) - \mathcal{T}_2 W_j(x)| = \dots \\ &= \left| \max_{u_1 \in U_1} \left(R(F(x,u_1),u_1) + \gamma W_j(F(x,u_1)) \right) - \max_{u_2 \in U_2} \left(R(F(x,u_2),u_2) + \gamma W_j(F(x,u_2)) \right) \right| \\ &\quad \text{rearranging,} \\ &= \left| \max_{u_1 \in U_1} \min_{u_2 \in U_2} \left(R(F(x,u_1),u_1) - R(F(x,u_2),u_2) \right) + \gamma \left(W_j(F(x,u_1)) - W_j(F(x,u_2)) \right) \right| \\ &\leq \max_{u_1 \in U_1} \min_{u_2 \in U_2} \left\{ \left| R(F(x,u_1),u_1) - R(F(x,u_2),u_2) \right| + \gamma \left| W_j(F(x,u_1)) - W_j(F(x,u_2)) \right| \right\} \\ &\quad \text{as } R \text{ is only a function of state,} \\ &\leq \max_{u_1 \in U_1} \min_{u_2 \in U_2} \left\{ \left| R(F(x,u_1)) - R(F(x,u_2)) \right| + \gamma \left| W_j(F(x,u_1)) - W_j(F(x,u_2)) \right| \right\} \\ &\quad \text{as } W_j, R \text{ are Lipschitz} \\ &\leq \max_{u_1 \in U_1} \min_{u_2 \in U_2} \left\{ L_R \|F(x,u_1) - F(x,u_2)\| + \gamma L_{W_j} \|F(x,u_1) - F(x,u_2)\| \right\} \\ &= (L_R + \gamma L_{W_j}) \max_{u_1 \in U_1} \min_{u_2 \in U_2} \left\{ \|F(x,u_1) - F(x,u_2)\| \right\} \\ &\quad \text{by the definition of Hausdorff distance,} \\ &\leq (L_R + \gamma L_{W_j}) d_S(\mathcal{R}_F(x,U_1), \mathcal{R}_F(x,U_2)) \end{aligned}$$

And hence

$$\|\mathcal{T}_1 W_j - \mathcal{T}_2 W_j\|_\infty \leq (L_R + \gamma L_{W_j}) \max_{x \in X} d_S(\mathcal{R}_F(x, U_1), \mathcal{R}_F(x, U_2)) \tag{A.24}$$

Therefore, the sequence Z_j satisfies

$$Z_{j+1} \leq \gamma Z_j + (L_R + \gamma L_{W_j}) \max_{x \in X} d_S(\mathcal{R}_F(x, U_1), \mathcal{R}_F(x, U_2)) \tag{A.25}$$

We note that all W_j inherit Lipschitz-ness from R and F, and as X is compact, they are also bounded. Therefore, $\mathrm{Lip}(W)$ *is a seminorm and therefore a continuous and bounded mapping, so by Lemma 4,*

$$\limsup_{j \to \infty} Z_j \leq \frac{(L_R + \gamma L_V)}{1 - \gamma} \max_{x \in X} d_S(\mathcal{R}_F(x, U_1), \mathcal{R}_F(x, U_2)) \tag{A.26}$$

where L_V is the Lipschitz constant of V_2^*. Note that using the other cross term results in the same analysis but with the Lipschitz constant of V_1^*, and hence write just "L_V". We additionally note that the exact value of L_V cannot be known without knowledge of the optimal solution. However, it can be upper bounded by the Lipschitz constant of the dynamics and reward functions.

As $\|\cdot\|_\infty$ is continuous and $\lim_{j\to\infty} V_j = V_1^*,\ \lim_{j\to\infty} W_j = V_2^*$, we note that the limit exists and conclude that

$$\|V_1^* - V_2^*\|_\infty \leq \frac{(L_R + \gamma L_V)}{1 - \gamma} \max_{x \in X} \left(d_S(\mathcal{R}_F(x, U_1), \mathcal{R}_F(x, U_2))\right). \tag{A.27}$$

Proof of Proposition 1

In preparation of for this lemma, we have some additional preliminaries.

The distance between two flows initialized at the same state and evolving under nonlinear dynamics and the corresponding local linearization is bounded.

Remark 6 *Consider the full nonlinear dynamics $F(x, u)$ and its linearization about an initial guess $(\bar{x}, \bar{u})$, $L(x, u) = F(\bar{x}, \bar{u}) + \nabla_x F|_{(\bar{x},\bar{u})}(x - \bar{x}) + \nabla_u F|_{\bar{x},\bar{u}}(u - \bar{u})$. For all states and inputs in an ε-ball centered at $(\bar{x}, \bar{u})$, the difference between the two functions is bounded as:*

$$\|F(x, u) - L(x, u)\| \leq \frac{1}{2}(L_{\nabla F})\varepsilon^2 \quad \forall (x, u) \in \bar{B}_\varepsilon((\bar{x}, \bar{u})) \tag{A.28}$$

where $L_{\nabla F}$ is the Lipschitz constant of the gradient of the dynamics. The proof follows directly from Taylor's Theorem.

Remark 7 is a structural observation that for deterministic MDPs, deciding an input separately along H timesteps is the same as deciding all H inputs at once.

Remark 7 *Consider a problem $\mathcal{M} = \langle X, U, F, R, D, K, \gamma \rangle$. We denote the H-horizon transcription of $\mathcal{M}$ as $\mathcal{M}^H = \left\langle X^H, U^H, F^H, R^H, D, K/H, \gamma^H \right\rangle$, where state $x_{[1:H]} = [x_1, \ldots, x_H]$ transfers according to input $u_{[H+1:2H]} = [u_{H+1}, \ldots, u_{2H}]$ as:*

$$F^H(x_{[1:H]}, u_{[H+1:2H]}) = \begin{bmatrix} F(x_H, u_{H+1}) \\ F(F(x_H, u_{H+1}), u_{H+2}) \\ \vdots \\ F(\ldots F(F(x_H, u_{H+1}), u_{H+2}), \ldots, u_{2H}) \end{bmatrix} \tag{A.29}$$

$$R^H(x_{[H+1:2H]}, u_{[H+1:2H]}) = \sum_{k=0}^{H-1} \gamma^k R(x_{H+k+1}, u_{H+k+1}) \tag{A.30}$$

$$s.t.\ x_{[H+1:2H]} = F^H(x_{[1:H]}, u_{[H+1:2H]}) \tag{A.31}$$

For notational simplicity, assume that K is an integer multiple of H; the extension to other cases is theoretically straightforward but notationally complicated. We note that $\mathcal{M}$ and $\mathcal{M}^H$ share the same optimal value function (noting existence of V^* when $\gamma < 1$). The proof is shown by writing out Equation (2.1) for both MDPs and observing they are the same problem, and hence share the same optimal solution.

The last preliminary shows the Discrete Algebraic Riccati Equation (DARE) controller induces a contracting system.

Lemma 5 *Consider a linear system, a dynamically feasible reference trajectory $\mathbf{z}^{\text{ref}}_{[H]}$, $u^{\text{ref}}_{[H]}$, and a feedback controller:*

$$\mathbf{z}_{k+1} = A\mathbf{z}_k + Bu_{k+1} + c \tag{A.32}$$

$$u_{k+1} = u^{\text{ref}}_{k+1} - \mathcal{K}(\mathbf{z}_k - \mathbf{z}^{\text{ref}}_k) \tag{A.33}$$

If the gain matrix $\mathcal{K}$ is selected as $(\Gamma_u + B^\top MB)^{-1}(B^\top MA)$, M solves the discrete algebraic Riccati equation (DARE) and $\Gamma_u \succ 0$, $\Gamma_x \succ 0$ then the system is contracting [2] at some rate $\alpha \in [0, 1)$.

The proof is given in Sec. A.1.

Now, we are ready to prove Prop. 1.

Proof 3 *The procedure of spectral expansion proceeds in three main steps (i) linearization of the full system, (ii) computation of the H-step controllable modes of the linearized system and (iii) feedback control to track the linear controllable modes with the nonlinear system.*

As such, we will compare the full H-step reachable set to the linearized reachable set, to its discretization into controllable modes, and to the nonlinear system tracking the controllable modes.

$$\mathcal{R}_{F^H}(x_0, U^H) \overset{\text{nonlinearity error}}{\longleftrightarrow} \mathcal{R}_{L^H}(x_0, U^H) \overset{\text{discrete eigenmodes}}{\longleftrightarrow} \{\mathbf{z}_H^i\}_{i=1}^{2n} \overset{\text{tracking linear trajectories}}{\longleftrightarrow} \{\mathbf{x}_H^i\}_{i=1}^{2n} \tag{A.34}$$

With the last being the reachable set under spectral expansion, and hence $\mathcal{R}_{F^H}(x_0, U_{\text{SETS}}) = \{\mathbf{x}_H^i\}_{i=1}^{2n}$. *As* d_S *is a metric,*

$$\begin{aligned} d_S(\mathcal{R}_{F^H}(x_0, U^H), \{x_i\}_{i=1}^{2n}) \leq\ & d_S(\mathcal{R}_{F^H}(x_0, U^H), \mathcal{R}_{L^H}(x_0, U^H)) \\ & + d_S(\mathcal{R}_{L^H}(x_0, U^H), \{\mathbf{z}_i\}_{i=1}^{2n}) \\ & + d_S(\{\mathbf{z}_i\}_{i=1}^{2n}, \{x_i\}_{i=1}^{2n}) \end{aligned} \tag{A.35}$$

We will bound each term of Equation (A.35) in turn.

Consider two trajectories with the same initial condition x_0, *driven by the same inputs, one with the full dynamics F and one with linearized dynamics L about a nominal trajectory starting at* x_0*:*

$$L_k(\mathbf{z}, u) = F(\bar{x}_k, \bar{u}_{k+1}) + \nabla_x F|_{(\bar{x}_k, \bar{u}_{k+1})}(\mathbf{z} - \bar{x}_k) + \nabla_u F|_{\bar{x}_k, \bar{u}_{k+1}}(u - \bar{u}_{k+1}) \tag{A.36}$$

$$= \underbrace{\nabla_x F|_{\bar{x}_k, \bar{u}_{k+1}}}_{A_k} \mathbf{z} + \underbrace{\nabla_u F|_{\bar{x}_k, \bar{u}_{k+1}}}_{B_k} u + \underbrace{F(\bar{x}_k, \bar{u}_{k+1}) - \nabla_x F|_{\bar{x}_k, \bar{u}_{k+1}} \bar{x}_k - F(\bar{x}_k, \bar{u}_{k+1})\bar{u}_{k+1}}_{c_k} \tag{A.37}$$

$$= A_k \mathbf{z} + B_k u + c_k \tag{A.38}$$

The corresponding dynamical systems are

$$x_{k+1} = F(x_k, u_{k+1}), \quad \forall k \in [0, H-1], \quad x_0 = x_0 \tag{A.39}$$

$$\mathbf{z}_{k+1} = L_k(x_k, u_{k+1}), \quad \forall k \in [0, H-1], \quad \mathbf{z}_0 = x_0 \tag{A.40}$$

Note that, $\forall k \in [0, H-1]$*:*

$$\begin{aligned}
\|x_{k+1} - \mathbf{z}_{k+1}\| &= \|F(x_k, u_{k+1}) - L_k(\mathbf{z}_k, u_{k+1})\| && \text{(A.41)}\\
&= \|F(x_k, u_{k+1}) - F(\mathbf{z}_k, u_{k+1}) + F(\mathbf{z}_k, u_{k+1}) - L_k(\mathbf{z}_k, u_{k+1})\| && \text{(A.42)}\\
&\quad \text{by the triangle inequality,} && \text{(A.43)}\\
&\leq \|F(x_k, u_{k+1}) - F(\mathbf{z}_k, u_{k+1})\| + \|F(\mathbf{z}_k, u_{k+1}) - L_k(\mathbf{z}_k, u_{k+1})\| && \text{(A.44)}\\
&\quad \text{as } F \text{ is Lipschitz, and by Remark 6} && \text{(A.45)}\\
&\leq L_F\|x_k - \mathbf{z}_k\| + \frac{1}{2}L_{\nabla F}\varepsilon^2 && \text{(A.46)}
\end{aligned}$$

Now as $x_0 = \mathbf{z}_0$*, the trajectories satisfy*

$$\|x_k - \mathbf{z}_k\| \leq \frac{(L_F)^k - 1}{L_F - 1}\left(\frac{1}{2}L_{\nabla F}\varepsilon^2\right) \quad \forall k \in [1, H] \tag{A.47}$$

Evaluating this inequality at timestep $k = H$ *and considering all pairs of linear and nonlinear trajectories yields a reachable set distance of:*

$$d_S(\mathcal{R}_{F^H}(x_0, U^H), \mathcal{R}_{L^H}(x_0, U^H)) \leq \frac{(L_F)^H - 1}{L_F - 1}\left(\frac{1}{2}L_{\nabla F}\varepsilon^2\right) \tag{A.48}$$

Now considering the linear reachable set $\mathcal{R}_{L^H}(x_0, U^H)$*, we note that*

$$\mathcal{R}_{L^H}(x_0, U^H) = \{\mathbf{z}_H \mid \mathbf{z}_H = \bar{\mathbf{z}}_H + C\mathbf{v}_{[H]} \text{ s.t. } \|\mathbf{v}_{[H]}\|_\infty \leq 1\} \tag{A.49}$$

where $\bar{\mathbf{z}}_H = L^H(x_0, \bar{u}_{[H]})$ *is the endpoint of the free linearized trajectory from* x_0 *and*

$$C = \left[\left(\textstyle\prod_{k=1}^{H-1} A_k\right)B_0S, \quad \left(\textstyle\prod_{k=2}^{H-1} A_k\right)B_1S, \quad \ldots, \quad A_{H-1}B_{H-2}S, \quad B_{H-1}S\right] \tag{A.50}$$

is the input-normalized controllability matrix of the linearized system. Here $\mathbf{v}_{[H]}$ *is a transformation of the input such that* $u_{[H]} \in U \implies \|\mathbf{v}_{[H]}\|_\infty \leq 1$*.*

Our algorithm uses a singular value decomposition of C *to extract the controllability information of our linear system. The left-singular vectors and singular values are the controllable directions and a measure of their controllability. Let the set* $\{\mathbf{z}_i\}_{i=1}^{2n}$ *be each left-singular vector of* C *times plus/minus its corresponding singular value, which we call the controllable modes of the system. We now quantify the distance between* $\mathcal{R}_{L^H}(x_0, U^H)$ *and* $\{\mathbf{z}_i\}_{i=1}^{2n}$*.* $\forall \mathbf{z}_H \in \mathcal{R}_{L^H}(x_0, U^H)$*, by Hölder's inequality,*

$$\|\mathbf{z}_H - \bar{\mathbf{z}}_H\|_2 \leq \|C\mathbf{v}_{[H]}\|_2 \leq \|C\|_2\|\mathbf{v}_{[H]}\|_\infty \leq \|C\|_2 \tag{A.51}$$

where $\|C\|_2 = \sigma_{\max}(C)$ *is the largest singular value of* C*.*

As $\sigma_{\max}(\cdot)$ is an induced matrix norm, we note that is submultiplicative. Using the block structure of C,

$$\sigma_{\max}(C) \leq \sum_{l=0}^{H-1} \sigma_{\max}\left((\prod_{k=l+1}^{H-1} A_k) B_l S \right) \tag{A.52}$$

$$\leq \sum_{l=0}^{H-1} \left[\left(\prod_{k=l+1}^{H-1} \sigma_{\max}(A_k) \right) \sigma_{\max}(B_l) \sigma_{\max}(S) \right] \tag{A.53}$$

which we bound by considering the maximum-singular value linearization over the state space. Noting the Lipschitz constant of F is defined as

$$L_F = \max_{x \in X, u \in U} \sigma_{\max}\left(\left[\nabla_x F|_{x,u} \; \nabla_u F|_{x,u} \right] \right) \tag{A.54}$$

then $\max_{x \in X, u \in U} \sigma_{\max}(\nabla_x F|_{x,u}) \leq L_F$ *and* $\max_{x \in X, u \in U} \sigma_{\max}(\nabla_u F|_{x,u}) \leq L_F$*, and consequently*

$$\sigma_{\max}(C) \leq \sum_{l=0}^{H-1} \left[\left(\prod_{k=l+1}^{H-1} L_F \right) L_F \sigma_{\max}(S) \right] = \sum_{l=0}^{H-1} \left[(L_F)^{H-l-1} L_F \sigma_{\max}(S) \right] \tag{A.55}$$

$$= \sum_{l=0}^{H-1} \left[(L_F)^{l} L_F \sigma_{\max}(S) \right] = \sigma_{\max}(S) L_F \left(\frac{(L_F)^H - 1}{L_F - 1} \right) \tag{A.56}$$

And now, noting that $\{\mathbf{z}_i\}_{i=1}^{2n} \subseteq \mathcal{R}_{L^H}(x_0, U^H)$*,*

$$d_S(\mathcal{R}_{L^H}(x_0, U^H), \{\mathbf{z}_i\}_{i=1}^{2n}) = \max_{x \in \mathcal{R}_{L^H}(x_0, U^H)} \min_i \|\mathbf{z}_i - x\| \tag{A.57}$$

$$= \max_{x \in \mathcal{R}_{L^H}(x_0, U^H)} \min_i \|\mathbf{z}_i - \bar{\mathbf{z}}_H + \bar{\mathbf{z}}_H - x\| \tag{A.58}$$

$$\leq \max_{x \in \mathcal{R}_{L^H}(x_0, U^H)} \min_i \|\mathbf{z}_i - \bar{\mathbf{z}}_H\| + \|x - \bar{\mathbf{z}}_H\| \tag{A.59}$$

$$\leq 2\sigma_{\max}(S) L_F \left(\frac{(L_F)^H - 1}{L_F - 1} \right) \tag{A.60}$$

Finally, an analogous argument to the first term gives a bound for the third term. For each controllable mode, we consider a desired trajectory to be the linear trajectory that terminates at $\mathbf{z}_i$. We fix a controller as a discrete algebraic Riccati controller tracking the desired trajectory $(\mathbf{z}^{\text{ref}}_{[H]}, u^{\text{ref}}_{[H]})$ to form autonomous nonlinear and linearized systems:

$$F_{cl}(x_k) = F(x_k, u^{\text{ref}}_{k+1} - \mathcal{K}_k(x_k - \mathbf{z}^{\text{ref}}_k)) \tag{A.61}$$

$$L_{cl,k}(\mathbf{z}_k) = F(\mathbf{z}_k, u^{\text{ref}}_{k+1} - \mathcal{K}_k(\mathbf{z}_k - \mathbf{z}^{\text{ref}}_k)) \tag{A.62}$$

Applying Lemma 5, this controller creates a contracting linear system with contraction rate α. Then, we treat the nonlinear system as a perturbed linear system, where the disturbance is the difference between the linear and nonlinear dynamics, and apply a robust contraction result [2] to bound the difference between the linear and nonlinear trajectory evolution. For all $k = 0, ..., H-1$

$$x_{k+1} = F_{cl}(x_k) = L_{cl,k}(x_k) + F_{cl}(x_k) - L_{cl,k}(x_k) \tag{A.63}$$

$$x_{k+1} - \mathbf{z}_{k+1} = L_{cl,k}(x_k) - L_{cl,k}(\mathbf{z}_k) + F_{cl}(x_k) - L_{cl,k}(x_k) \tag{A.64}$$

$$\|x_{k+1} - \mathbf{z}_{k+1}\| \leq \alpha\|x_k - \mathbf{z}_k\| + F_{cl}(x_k) - L_{cl,k}(x_k) \tag{A.65}$$

$$\|x_{k+1} - \mathbf{z}_{k+1}\| \leq \alpha\|x_k - \mathbf{z}_k\| + \frac{1}{2}(L_{\nabla F_{cl}})\varepsilon^2 \tag{A.66}$$

and hence for all $k = 1, ..., H$:

$$\|x_k - \mathbf{z}_k\| \leq \alpha^k\|x_0 - \mathbf{z}_0^{\text{ref}}\| + \frac{1-\alpha^k}{1-\alpha}\left(\frac{1}{2}(L_{\nabla F_{cl}})\varepsilon^2\right) \tag{A.67}$$

noting that F_{cl} inherits Lipschitz gradients from F, and furthermore $L_{\nabla F_{cl}} \leq L_{\nabla F}$. As the trajectories start from the same initial condition, we are left with only the second term. Hence the nonlinear rollout error associated with tracking $(\mathbf{z}_{[H]}^{\text{ref}}, u_{[H]}^{\text{ref}})$ is bounded as

$$\|x_H - \mathbf{z}_H\| \leq \frac{1-\alpha^H}{1-\alpha}\left(\frac{1}{2}(L_{\nabla F_{cl}})\varepsilon^2\right) \tag{A.68}$$

As this bound holds for each controllable mode endpoint $\mathbf{z}_i$,

$$d_S(\{\mathbf{z}_i\}_{i=1}^{2n}, \{x_i\}_{i=1}^{2n}) \leq \frac{1}{2}(L_{\nabla F_{cl}})\varepsilon^2\frac{1-\alpha^H}{1-\alpha} \tag{A.69}$$

Substituting into (A.35) completes the proof.

Remark 8 *Although the term $(L_F)^H$ is discouraging, it arises from a linear approximation of a nonlinear dynamical system and it may be unavoidable because, in general, computing a nonlinear reachable set is NP-Hard. Despite its exponential growth, we can control this term when F is derived as an Euler integration over Δt of a continuous dynamical system $\dot{x} = f(x, u)$. When f is Lipschitz with constant L_f, then $L_F = 1 + \Delta t L_f$. For a fixed H, shrinking Δt controls the growth of this term. For a fixed effective planning horizon (constant $H\Delta t$), increasing Δt and decreasing H proportionally shrinks this term.*

Proof of Lemma 5:

Proof 4 *Combine the system and controller to write the closed loop system and its differential dynamics:*

$$\mathbf{z}_{k+1} = A\mathbf{z}_k + Bu_{k+1}^{\text{ref}} - B\mathcal{K}(\mathbf{z}_k - \mathbf{z}_k^{\text{ref}}) + c \tag{A.70}$$

$$\delta\mathbf{z}_{k+1} = \underbrace{(A - B\mathcal{K})}_{A_{cl}}\delta\mathbf{z}_k \tag{A.71}$$

Recall the definition of discrete-time contraction from [3]: A discrete-time system is contracting with time-invariant metric M iff

$$A_{cl}^\top M A_{cl} - \alpha^2 M \preceq 0 \tag{A.72}$$

where A_{cl} is the closed loop dynamics of the differential system and $\alpha \in [0, 1)$ is the contraction rate.

Compute the left-hand-side of the contraction condition, and we will seek to show it is negative definite to prove contraction (A.72)*:*

$$\text{LHS} = (A - B\mathcal{K})^\top M(A - B\mathcal{K}) - \alpha^2 M \tag{A.73}$$

$$= A^\top MA - A^\top MB\mathcal{K} - \mathcal{K}^\top B^\top MA + \mathcal{K}^\top B^\top MB\mathcal{K} - \alpha^2 M \tag{A.74}$$

Manipulate this term, $\mathcal{K}^\top B^\top MB\mathcal{K}$, by plugging in the definition of $\mathcal{K}$, add/subtracting $\mathcal{K}^\top\Gamma_u\mathcal{K}$ and grouping terms:

$$\mathcal{K}^\top B^\top MB\mathcal{K} = \mathcal{K}^\top B^\top MB(\Gamma_u + B^\top MB)^{-1}(B^\top MA) \tag{A.75}$$

$$= \mathcal{K}^\top B^\top MB(\Gamma_u + B^\top MB)^{-1}(B^\top MA) + \mathcal{K}^\top\Gamma_u\mathcal{K} - \mathcal{K}^\top\Gamma_u\mathcal{K} \tag{A.76}$$

$$= \mathcal{K}^\top(\Gamma_u + B^\top MB)(\Gamma_u + B^\top MB)^{-1}(B^\top MA) - \mathcal{K}^\top\Gamma_u\mathcal{K} \tag{A.77}$$

$$= \mathcal{K}^\top B^\top MA - \mathcal{K}^\top\Gamma_u\mathcal{K} \tag{A.78}$$

Plug back into LHS:

$$\text{LHS} = A^\top MA - A^\top MB\mathcal{K} - \mathcal{K}^\top\Gamma_u\mathcal{K} - \alpha^2 M \tag{A.79}$$

Recall the definition of the discrete algebraic Riccati equation, DARE$(A, B, \Gamma_x, \Gamma_u)$ [4]:

$$M = A^\top MA - A^\top MB(\Gamma_u + B^\top MB)^{-1}(B^\top MA) + \Gamma_x \tag{A.80}$$

Let M solve DARE$(A, B, \Gamma_x, \Gamma_u)$. Plug into LHS:

$$\text{LHS} = -\Gamma_x - \mathcal{K}^\top\Gamma_u\mathcal{K} + (1 - \alpha^2)M \tag{A.81}$$

Because $\mathcal{K}^\top\Gamma_u\mathcal{K} \succeq 0$, if we select Γ_x to be sufficiently large, the system is contracting with rate α.

Proof of Theorem 1

Proof 5 *As in Remark 7, let $\mathcal{M}^H$ denote the H-horizon transcription of $\mathcal{M}$. The optimal value function of $\mathcal{M}^H$ and $\mathcal{M}$ are the same:*

$$(V^H)^*(x) = V^*(x) \quad \forall x \in X \tag{A.82}$$

The algorithm SETS constructs and solves a new MDP

$$\mathcal{M}_{\text{SETS}} = \left\langle X, U_{\text{SETS}}, F^H, R^H, V, K/H, \gamma^H \right\rangle$$

using UCT to navigate the decision tree. We note that the branching factor of $\mathcal{M}_{\text{SETS}}$ is finite and bounded by twice the state dimension n. As $\gamma^H < 1$, there exists a unique optimal value function V^*_{SETS} associated with $\mathcal{M}_{\text{SETS}}$. Let $\hat{V}(x_0, \ell)$ denote the value estimate formed by the l^{th} iteration of UCT running on $\mathcal{M}_{\text{SETS}}$. By Lemma 1, $\hat{V}(x_0, l)$ satisfies

$$V^*_{\text{SETS}}(x_0) - \mathbb{E}[\hat{V}(x_0, \ell)] \leq O\left(\frac{(2n)(K/H)\log(\ell) + (2n)^{(K/H)}}{\ell}\right) \tag{A.83}$$

By Lemma 2 and Proposition 1, as $\mathcal{M}_{\text{SETS}}$ and $\mathcal{M}^H$ share the same problem information apart from the input set,

$$\|(V^H)^* - V^*_{\text{SETS}}\|_\infty \leq \frac{L_R + \gamma^H L_V}{1-\gamma^H}\left(\frac{1}{2}(L_{\nabla F})\varepsilon^2 \frac{(L_F)^H - 1}{L_F - 1}\right. \tag{A.84}$$

$$\left. + 2\sigma_{\max}(S)L_F\left(\frac{(L_F)^H - 1}{L_F - 1}\right) + \frac{1}{2}(L_{\nabla F_{cl}})\varepsilon^2 \frac{1-\alpha^H}{1-\alpha}\right) \tag{A.85}$$

The triangle inequality yields the relationship:

$$V^*(x_0) - \mathbb{E}[\hat{V}(x_0, \ell)] = \left(V^*(x_0) - (V^H)^*(x_0)\right) + \left((V^H)^*(x_0) - V^*_{\text{SETS}}(x_0)\right) \tag{A.86}$$

$$+ \left(V^*_{\text{SETS}}(x_0) - \mathbb{E}[\hat{V}(x_0, l)]\right) \tag{A.87}$$

$$\leq 0 + \frac{L_R + \gamma^H L_V}{1-\gamma^H}\left(\frac{1}{2}(L_{\nabla F})\varepsilon^2 \frac{(L_F)^H - 1}{L_F - 1} + 2\sigma_{\max}(S)L_F\left(\frac{(L_F)^H - 1}{L_F - 1}\right)\right. \tag{A.88}$$

$$\left. + \frac{1}{2}(L_{\nabla F_{cl}})\varepsilon^2 \frac{1-\alpha^H}{1-\alpha}\right) + O\left(\frac{(2n)(K/H)\log(\ell) + (2n)^{(K/H)}}{\ell}\right) \tag{A.89}$$

Noting the feedback control term is bounded as:

$$\frac{1}{2}(L_{\nabla F_{cl}})\varepsilon^2 \frac{1-\alpha^H}{1-\alpha} \leq \frac{1}{2}(L_{\nabla F_{cl}})\varepsilon^2 \frac{1}{1-\alpha} \tag{A.90}$$

rearrangement with

$$C_0 = \frac{L_R}{L_V},\ C_1 = \frac{L_V(L_{\nabla F}\varepsilon^2 + 4\sigma_{\max}(S)L_F)}{2(L_F - 1)},\ C_2 = L_f, \tag{A.91}$$

$$C_3 = \frac{L_V L_{\nabla F_{cl}}\varepsilon^2}{2(1-\alpha)} - \frac{L_V(L_{\nabla F}\varepsilon^2 + 4\sigma_{\max}(S)L_F)}{2(L_F - 1)} \tag{A.92}$$

completes the proof.

A.2 MCTS for Belief-Space Planning

Lemma 6 (Equivalent POMDP reformulation) *If a global optimal solution exists to the constrained safe active fault estimation problem, Definition 6, then the solution of the following POMDP is also a global optimal solution of Definition 6 in the discretization limit of state and observation space:*

$$\begin{aligned}
&\langle Q, U, Y, R_{h,\alpha}, T, Z\rangle,\\
&T(q_k, u_k, q_{k+1}) = \mathbb{P}\left(\begin{bmatrix} x_k \\ \phi_k \end{bmatrix} = \begin{bmatrix} f(x_{k-1}) + B(x_{k-1})\left((\mathbb{I} - \Phi_B)u_k + \Phi_{B,\mathbb{1}}\right) + w_k \\ \phi_{k-1} \end{bmatrix}\right),\\
&Z(q_k, u_k, y_k) = \mathbb{P}\left(y_k = (\mathbb{I} - \Phi_C)Cx_k + \Phi_{C,\mathbb{1}} + v_k\right)
\end{aligned} \tag{S8}$$

where $Q = X \times \Phi$, U, Y are as in Definition 6, $R_{h,\alpha}$ is given by Eq. (3.21)*, and the belief updates are given by Eq.* (3.12)*. Further, when $R_{h,\alpha}$ is replaced by $\tilde{R}_{h,\alpha}(b_k)$ given by Eq.* (3.29)*, then the global optimal solution of this POMDP is a global optimal solution of the conservative safe active fault estimation problem (Definition 7).*

Proof 6 *By assumption the policy $\pi^*(b_0)$, a global optimal solution to the original problem (Definition 6), 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 $\pi^*_{h,\alpha}(b_0)$, another optimal solution to Definition 6 satisfying:*

$$\pi^*_{h,\alpha}(b_0) = \arg\max_{\pi\in\Pi} V^{\pi}_{h,\alpha}(b_0) \tag{3.23}$$

So we have the following

$$V^{\pi^*_{h,\alpha}}_{h,\alpha}(b_0) \geq V^{\pi}_{h,\alpha}(b_0) \quad \forall \pi \in \Pi \tag{S9}$$

Because the POMDP given by Eq. (S8) *also uses $R_{h,\alpha}$ given by Eq.* (3.21) *as a reward, it shares the value function given by Eq.* (3.22) *with the reformulation*

given by Theorem 2. Therefore, $\pi^*_{h,\alpha}(b_0)$ and $\pi^*(b_0)$ maximize the value function of Eq. (S8)*, so, by definition, the solution of Definition 6 is an optimal policy of Eq.* (S8) *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.*

Theorem 9 (Equivalent unbounded reformulation) *If a global optimal solution exists to the constrained safe active fault estimation problem, Definition 6, then the solution of the following unconstrained problem with a transformed value function given by Eq.* (3.22)*, is also a global optimal solution of Definition 6:*

$$\pi^*_{h,\alpha}(b_0) = \arg\max_{\pi\in\Pi} V^{\pi}_{h,\alpha}(b_0) \tag{3.23}$$

Proof 7 *The equivalence of the problems is shown if the optimal policy of the original problem given by Definition 6, Eq.* (3.17) *has the same value as the optimal policy of the reformulated problem given by Eq.* (3.23)*. The reformulated problem is constructed such that any policy resulting in an expected α-safe trajectory (Definition 5) has a minimum expected cumulative reward of Kr_0, which is higher than the maximum expected cumulative reward of an expected unsafe trajectory, $K-1$, as $Kr_0 = \frac{K^2}{K+1} > \frac{K^2-1}{K+1} = K-1$.*

$$\mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi, b_0] = 1 \ \forall k \iff V^{\pi}_{h,\alpha}(b_0) \geq Kr_0 \tag{S1}$$

$$\exists k: \ \mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi, b_0] \neq 1 \iff V^{\pi}_{h,\alpha}(b_0) < Kr_0 \tag{S2}$$

By assumption, the policy $\pi^*(b_0)$, a global optimal solution to Definition 6, exists and is feasible. From the constraints of Eq. (3.17)*, this solution must satisfy:*

$$\mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi^*, b_0] = 1 \ \forall k \implies V^{\pi^*}_{h,\alpha}(b_0) \geq Kr_0 \tag{S3}$$

Since $\pi^*_{h,\alpha}(b_0)$ is the optimal solution to the reformulation given by Eq. (3.23)*:*

$$V^{\pi^*_{h,\alpha}}_{h,\alpha}(b_0) \geq V^{\pi}_{h,\alpha}(b_0), \forall\pi; \ \ V^{\pi^*_{h,\alpha}}_{h,\alpha}(b_0) \geq Kr_0 \implies \mathbb{E}[\mathbb{1}_{\mathcal{B}_{h,\alpha}}(b_k) \mid \pi^*_{h,\alpha}, b_0] = 1 \forall k \tag{S4}$$

So $\pi^*_{h,\alpha}(b_0)$ satisfies the original problem constraints (Definition 6). 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 6), the transformed objective is an affine transformation and is monotonic, so with Eq. (S4)*:*

$$V^*(b_0) \geq V^{\pi^*_{h,\alpha}}(b_0) \implies V^{\pi^*}_{h,\alpha}(b_0) \geq V^{\pi^*_{h,\alpha}}_{h,\alpha}(b_0) \implies V^{\pi^*}_{h,\alpha}(b_0) = V^{\pi^*_{h,\alpha}}_{h,\alpha}(b_0) \tag{S5}$$

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

Theorem 10 (Conservative sampling bound) *For $M > 2$, a belief $b(x)$, safety function h, $\hat{\mu}_h, \hat{\sigma}_h$ defined according to Eq.* (3.25)*, and $\hat{\mu}_h \geq \hat{\sigma}_h$; satisfying the approximate safety condition of Eq.* (3.20) *indicates that the belief is conservatively α-safe.*

$$\frac{1}{M+1}\left\lfloor \frac{M+1}{M}\left(\frac{\hat{\sigma}_h^2(M-1)}{\hat{\mu}_h^2}+1\right)\right\rfloor \leq 1-\alpha \implies b \in \mathcal{B}_{h,\alpha} \tag{3.20}$$

Proof 8 *Defining the physical state associated with the belief as: $Z \sim b(x)$, a random variable, the condition for α-safety is $\mathbb{P}(h(x) \geq 0) \geq \alpha \implies \mathbb{P}(h(x) < 0) \leq 1-\alpha$. Adding and subtracting the empirical mean and upper bounding the one sided tail probability with a two-sided condition we have:*

$$\mathbb{P}(h(x) < 0) = \mathbb{P}(h(x_i) - \hat{\mu}_h < -\hat{\mu}_h) \leq \mathbb{P}(|h(x) - \hat{\mu}_h| > \hat{\mu}_h) \tag{S6}$$

Using Eq. (3.24)*, and choosing $\lambda = \hat{\mu}_h/\hat{\sigma}_h$ we have:*

$$\mathbb{P}(|h(x) - \hat{\mu}_h| > \hat{\mu}_h) \leq \frac{1}{M+1}\left\lfloor \frac{M+1}{M}(\frac{\hat{\sigma}_h^2(M-1)}{\hat{\mu}_h^2}+1)\right\rfloor \tag{S7}$$

Combining Eqs. (S6) *and* (S7)*, we arrive at the desired result.*

Lemma 7 (Conservative α-safe set) *For a 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:*

$$\frac{\sigma_h^2}{\mu_h^2} \leq 1-\alpha \iff b \in \tilde{\mathcal{B}}_{h,\alpha} \tag{3.26}$$

Proof 9 *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.* (3.20) *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 $\tilde{\mathcal{B}}_{h,\alpha} \subseteq \mathcal{B}_{h,\alpha}$.*

Theorem 11 (Problem reformulation equivalence) *If an admissible policy, $\pi(b_0)$, to the safe active fault estimation problem (Definition 6) exists and satisfies:*

$$\mathbb{E}[\mathbb{1}_{\tilde{\mathcal{B}}_{h,\alpha}}(b_k) \mid \pi, b_0] = 1 \ \forall k \tag{3.30}$$

where $\tilde{\mathcal{B}}_{h,\alpha}$ is given by Lemma 3, then an optimal policy, $\tilde{\pi}^*_{h,\alpha}(b_0)$, to the conservative safe active fault estimation problem (Definition 7) is a sub-optimal solution of Definition 6 constrained to $\tilde{\mathcal{B}}_{h,\alpha}$. Further, if an optimal policy, $\pi^*(b_0)$, to Definition 6 exists and satisfies Eq. (3.30)*, $\tilde{\pi}^*_{h,\alpha}(b_0)$ is an optimal solution to Definition 6.*

Proof 10 *We start with the second claim. From Theorem 2, $\tilde{\pi}^*_{h,\alpha}(b_0)$, is optimal on the the safe active fault estimation problem (Definition 6) restricted to $\tilde{\mathcal{B}}_{h,\alpha}$. As $\mathbb{E}[\mathbb{1}_{\tilde{\mathcal{B}}_{h,\alpha}}(b_k) \mid \pi^*, b_0] = 1 \; \forall k$, relaxing the restriction to $\mathcal{B}_{h,\alpha}$ does not change the optimal value, so $\tilde{\pi}^*_{h,\alpha}(b_0)$ is an optimal solution to Definition 6.*

For the first claim, when a feasible policy of Definition 6 generates expected beliefs in $\tilde{\mathcal{B}}_{h,\alpha}$ ($\exists \pi$ satisfying Eq. (3.30)*), 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.*

Here we present an equivalent reformulation to make s-FEAST compatible with existing search methods, then use this compatibility to inherit their convergence guarantees and prove the optimality of s-FEAST.

Theorem 12 (Optimality of s-FEAST) *Let μ 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 7). In the limit of $M \to \infty$, the value of these policies converge:*

$$\lim_{N\to\infty} \left(\tilde{V}^{\mu}_{h,\alpha}(b_0) - \tilde{V}^{*}_{h,\alpha}(b_0) \right) \to 0 \tag{3.31}$$

with convergence rate $O(\log N/N)$. Further, if an optimal policy, $\pi^*(b_0)$, to Definition 6 exists and satisfies Eq. (3.30)*, $V^{\mu}(b_0)$ converges to $V^*(b_0)$.*

Proof 11 *From Lemma 3 we have that in the limit of $M \to \infty$, $\tilde{\mathcal{B}}_{h,\alpha}$ is the set for which Eq.* (3.26) *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 7) with the stated convergence rate, we achieved both the claimed results. To show s-FEAST solves Definition 7, we note that Definition 7 can be equivalently reformulated as a POMDP by Lemma 6. To solve the equivalent POMDP, s-FEAST employs the marginalized filter given by Eq.* (3.19) *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 (46) and inherit the convergence rate from their Theorem 1.*

A.3 Global to Local Learning

Theorem 13 *For the single integrator dynamics* (5.3)*, the safety defined by* (5.13) *is guaranteed under the control law* (5.6) *with the following* $b(o^i)$ *and* $\pi(o^i)$ *for a scalar gains* $k_p > 0$ *and* $k_c > 0$*:*

$$b(o^i) = -k_p \nabla_{p^i} \Psi^i(o^i) \tag{A.93}$$

$$\alpha_\pi(o^i) = \begin{cases} \frac{(k_p - k_c)\|\nabla_{p^i}\Psi^i(o^i)\|^2}{k_p\|\nabla_{p^i}\Psi^i(o^i)\|^2 + |\langle \nabla_{p^i}\Psi^i(o^i), \pi(o^i)\rangle|} & \Delta_h(o^i) < 0 \\ 1 & else \end{cases} \tag{A.94}$$

with $\Delta_h(o^i) = \min_{j \in \mathcal{N}^i} h(\overline{p}^{ij}) - \Delta_r$*, and* $\nabla_{p^i}\Psi^i$ *as in* (A.97)*.*

Proof 12 *We prove global safety by showing boundedness of* $\Psi^i(o^i)$*, because a bounded* $\Psi(x)$ *implies no safety violation. Since* $\Psi(x)$ *is a sum of functions* $\Psi^i(o^i)$*, it suffices to show that all* $\Psi^i(o^i)$ *are bounded. To prove the boundedness of* $\Psi^i(o^i)$*, we use a Lyapunov method. First, we note that* $\Psi^i(o^i)$ *is an appropriate positive Lyapunov function candidate as* $\Psi^i(o^i) > 0 \ \forall o^i$*, using the fact that* $-\log(x) \in (0, \infty), \ \forall x \in (0, 1)$*. Second, we show* $x \in \partial X_s^i \implies \dot{\Psi}^i(o^i) < 0$*, where the* boundary-layer *domain* $\partial X_s^i \subset X$ *is defined as:*

$$\partial X_s^i = \{x \mid 0 < \min_{j \in \mathcal{N}^i} h(\overline{p}^{ij}) < \Delta_r\}. \tag{A.95}$$

This result implies that upon entering the boundary-layer of the safe set, ∂X_s^i*, the controller will push the system back into the interior of the safety set,* $X \setminus \partial X_s^i$*.*

We take the time derivative of Ψ^i *along the system dynamics, and plug the single integrator dynamics* (5.3) *and controller* (5.6) *into* $\dot{\Psi}^i$*:*

$$\begin{aligned} \dot{\Psi}^i &= \langle \nabla_x \Psi^i, \dot{x} \rangle = \sum_{i=1}^N \langle \nabla_{x^i} \Psi^i, \dot{x}^i \rangle = \sum_{i=1}^N \langle \nabla_{p^i} \Psi^i, u^i \rangle \\ &= \sum_{i=1}^N \langle \nabla_{p^i} \Psi^i, \alpha_\pi \pi + (1 - \alpha_\pi) b \rangle, \end{aligned} \tag{A.96}$$

$$\text{where } \nabla_{p^i}\Psi^i = \sum_{j \in \mathcal{N}^i} \frac{\overline{p}^{ij}}{\|\overline{p}^{ij}\|(\|\overline{p}^{ij}\| - r_{\text{safe}})}. \tag{A.97}$$

From here, establishing that any element of the sum is negative when $x \in \partial X_s^i$ *implies the desired result. Expanding an arbitrary element at the i index with the definition of b:*

$$\begin{aligned} &= -k_p \|\nabla_{p^i}\Psi^i\|^2 + \alpha_\pi(\langle \nabla_{p^i}\Psi^i, \pi \rangle + k_p \|\nabla_{p^i}\Psi^i\|^2) \\ &\leq -k_p \|\nabla_{p^i}\Psi^i\|^2 + \pi(|\langle \nabla_{p^i}\Psi^i, \pi \rangle| + k_p \|\nabla_{p^i}\Psi^i\|^2) \end{aligned}$$

By plugging π from (5.18) *into* (A.98)*, we arrive at the following:*

$$\langle \nabla_{p^i} \Psi^i, \pi \alpha_\pi + (1 - \alpha_p i) b \rangle \leq -k_c \|\nabla_{p^i} \Psi^i\|^2 \tag{A.98}$$

Thus, every element is strictly negative, unless $\nabla_{p^i} \Psi^i$ equals zero; caused either by reaching the safety equilibrium of the system at $h(\overline{p}^{ij}) \geq 1, \forall j \in \mathcal{N}^i$, or in the case of deadlock between robots.

Theorem 14 *For the double integrator dynamics given in* (5.3)*, the safety defined by* (5.13) *is guaranteed under control law* (5.6) *for the barrier-controller and the gain defined as:*

$$\begin{aligned} b &= -k_v(v^i + k_p \nabla_{p^i} \Psi^i) - k_p \frac{d}{dt} \nabla_{p^i} \Psi^i - k_p \nabla_{p^i} \Psi^i, \\ \alpha_\pi &= \begin{cases} \frac{a_1 - k_c(k_p \Psi^i + \frac{1}{2}\|v-k\|^2)}{a_1 + |a_2|} & \Delta_h(o^i) < 0 \\ 1 & else \end{cases} \\ a_1 &= k_v \|v^i + k_p \nabla_{p^i} \Psi^i\|^2 + k_p^2 \|\nabla_{p^i} \Psi^i\|^2, \\ a_2 &= \langle v^i, k_p \nabla_{p^i} \Psi^i \rangle + \langle v^i + k_p \nabla_{p^i} \Psi^i, \pi + k_p \frac{d}{dt} \nabla_{p^i} \Psi^i \rangle, \end{aligned} \tag{A.99}$$

where $k_p > 0$, $k_c > 0$, and $k_v > 0$ are scalar gains, Δ_h is defined as in Theorem 7, $\frac{d}{dt} \nabla_{p^i} \Psi^i$ is defined in (A.105) *and the dependency on the observation is suppressed for legibility.*

Proof 13 *We take the same proof approach as in Theorem 7 to show boundedness of ψ^i. We define a Lyapunov function, $\mathbb{V}$ augmented with a backstepping term:*

$$\mathbb{V} = k_p \Psi^i + \frac{1}{2} \|v - k\|^2, \tag{A.100}$$

where v is the stacked velocity vector, $v = [v^1; \ldots; v^N]$ and k is the stacked nominally stabilizing control, $k = -k_p[\nabla_{p^1} \Psi^1; \ldots; \nabla_{p^N} \Psi^N]$. For the same reasoning as the previous result, $\mathbb{V}$ is a positive function, and thus an appropriate Lyapunov candidate. Taking the derivative along the system dynamics (5.3)*:*

$$\dot{\mathbb{V}} = \sum_{i=1}^{N} k_p \langle \nabla_{x^i} \Psi^i, \dot{x}^i \rangle + \langle v^i - k^i, u^i - \dot{k}^i \rangle \tag{A.101}$$

$$= \sum_{i=1}^{N} k_p \langle \nabla_{p^i} \Psi^i, v^i \rangle + \langle v^i - k^i, u^i - \dot{k}^i \rangle. \tag{A.102}$$

From here, establishing that an arbitrary element of the sum is negative when $x \in \partial X_s^i$ implies the desired result. We rewrite the first inner product in this expression as:

$$\begin{aligned}\langle \nabla_{p^i} \Psi^i, v^i \rangle &= \langle \nabla_{p^i} \Psi^i, k^i \rangle + \langle \nabla_{p^i} \Psi^i, (v^i - k^i) \rangle \\ &= -k_p \|\nabla_{p^i} \Psi^i\|^2 + \langle \nabla_{p^i} \Psi^i, (v^i - k^i) \rangle. \end{aligned} \tag{A.103}$$

Next, we expand the second inner product of (A.102)*, and plug k^i,u^i, and b into* (A.102)*:*

$$\begin{aligned}&\langle v^i + k_p \nabla_{p^i} \Psi^i, u^i + k_p \frac{d}{dt} \nabla_{p^i} \Psi^i \rangle \\ &= -k_v \|v^i + k_p \nabla_{p^i} \Psi^i\|^2 - k_p \langle \nabla_{p^i} \psi^i, v^i + k_p \nabla_{p^i} \Psi^i \rangle \\ &\qquad + \pi \langle v^i + k_p \nabla_{p^i} \Psi^i, \alpha_\pi - b \rangle \end{aligned} \tag{A.104}$$

$$\begin{aligned}&\textit{where } \frac{d}{dt} \nabla_{p^i} \Psi^i = \sum_{j \in \mathcal{N}^i} \frac{v^i}{\|\overline{p}^{ij}\|(\|\overline{p}^{ij}\| - r_{\text{safe}})} \\ &- \frac{\langle p^i, v^i \rangle p^i}{\|\overline{p}^{ij}\|(\|\overline{p}^{ij}\| - r_{\text{safe}})(\|\overline{p}^{ij}\|^2 + (\|\overline{p}^{ij}\| - r_{\text{safe}})^2)} \end{aligned} \tag{A.105}$$

Combining both terms back into the expression:

$$\begin{aligned}&k_p \langle \nabla_{p^i} \Psi^i, v^i \rangle + \langle v^i - k^i, u^i - \dot{k}^i \rangle = -k_p^2 \|\nabla_{p^i} \Psi^i\|^2 \\ &-k_v \|v^i + k_p \nabla_{p^i} \Psi^i\|^2 + \pi \langle v^i + k_p \nabla_{p^i} \Psi^i, \alpha_\pi - b \rangle \end{aligned} \tag{A.106}$$

Expanding the last term with the definition of b, and upper bounding it:

$$\begin{aligned}&\langle v^i + k_p \nabla_{p^i} \Psi^i, \pi - b \rangle \\ &= (k_v \|v^i + k_p \nabla_{p^i} \Psi^i\|^2 + k_p^2 \|\nabla_{p^i} \Psi^i\|^2) \\ &+ (\langle v^i, k_p \nabla_{p^i} \Psi^i \rangle + \langle v^i + k_p \nabla_{p^i} \Psi^i, \pi + k_p \frac{d}{dt} \nabla_{p^i} \Psi^i \rangle) \\ &= a_1 + a_2 \leq a_1 + |a_2| \end{aligned} \tag{A.107}$$

where the terms in parentheses are grouped into a_1, a_2 to improve legibility. By plugging α_π from (5.19) *into* (A.106)*, we arrive at $\dot{\mathbb{V}} = \sum_{i=1}^N -k_c \mathbb{V}$, which results in exponential stability that guarantees the system will remain safe by pushing it towards a safety equilibrium where $h(\overline{p}^{ij}) \geq 1, \forall j \in \mathcal{N}^i$. It also makes the system robust to disturbances [5].*

BIBLIOGRAPHY

[1] Andrew Stuart and Anthony R Humphries. *Dynamical Systems and Numerical Analysis*. Vol. 2. Cambridge University Press, 1998.

[2] Winfried Lohmiller and Jean-Jacques E. Slotine. "On Contraction Analysis for Non-linear Systems". In: *Automatica* 34.6 (1998), pp. 683–696. DOI: 10.1016/S0005-1098(98)00019-3. URL: https://doi.org/10.1016/S0005-1098(98)00019-3.

[3] Hiroyasu Tsukamoto, Soon-Jo Chung, and Jean-Jacques E. Slotine. "Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview". In: *Annual Reviews in Control* 52 (2021), pp. 135–169. DOI: 10.1016/J.ARCONTROL.2021.10.001. URL: https://doi.org/10.1016/j.arcontrol.2021.10.001.

[4] D.E. Kirk. *Optimal Control Theory: An Introduction*. Dover Books on Electrical Engineering. Dover Publications, 2012. ISBN: 9780486135076. URL: https://books.google.com/books?id=onuH0PnZwV4C.

[5] Hassan K Khalil. *Nonlinear systems*. Upper Saddle River, NJ: Prentice-Hall, 2002. ISBN: 978-0130673893. URL: https://cds.cern.ch/record/1173048.

Appendix B

IMPLEMENTATION DETAILS

B.1 MCTS for Dynamics

Problem Data and Equations of Motion

Quadrotor

The state and action space are specified with the following parameters:

$$X = \begin{bmatrix} -1.3 & -1.3 & -3.2 & -6.0 & -6.0 & -6.0 & -0.8 & -0.8 & -10.0 & -5.0 & -5.0 & -5.0 \\ 1.3 & 1.3 & -2.2 & 6.0 & 6.0 & 6.0 & 0.8 & 0.8 & 10.0 & 5.0 & 5.0 & 5.0 \end{bmatrix}^\top \tag{B.1}$$

$$U = \begin{bmatrix} 0.0 & -0.012 & -0.012 & -0.002 \\ 12.0 & 0.012 & 0.012 & 0.002 \end{bmatrix}^\top \tag{B.2}$$

For the dynamical model, we opt for the convention from [13] for a 6DOF model of the quadrotor, rather than standard compact notation [14], because it will be reused for the glider model in Sec. B.1. The dynamics of the nominal physical system are the following:

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{bmatrix} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{\mathrm{m}} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6 (p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{bmatrix} + \begin{bmatrix} \Gamma_3 l + \Gamma_4 n \\ \frac{1}{J_y} m \\ \Gamma_4 l + \Gamma_8 n \end{bmatrix}$$

where (i) s, c, t are shorthand notation for sin, cos, and tan, (ii) m, $\{\Gamma_i\}_{i=1}^8$ are mass and moment of inertia constants, (iii) f_x, f_y, f_z are the external forces in body frame and n, m, l are the external torques.

Tracked Vehicle

The nominal tracked vehicle dynamics are the following:

$$F\left(\begin{bmatrix} x \\ y \\ \theta \\ v \\ \omega \end{bmatrix}, \begin{bmatrix} v_d \\ \omega_d \end{bmatrix}\right) = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \omega \end{bmatrix} + \Delta t \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ \frac{1}{\tau_v}(-v + v_d) \\ \frac{1}{\tau_\omega}(-\omega + \omega_d) \end{bmatrix} \tag{B.3}$$

for parameters $\Delta t = 0.1$, $\tau_v = 0.2, \tau_\omega = 0.15$.

The state space and action space are specified with the following parameters:

$$X = \begin{bmatrix} -100 & -100 & -100 & -10\pi & -1.8 & -1.5 \\ 100 & 100 & 100 & 10\pi & 1.8 & 1.5 \end{bmatrix}^\top \quad U = \begin{bmatrix} -1.0, & -1.0 \\ 1.0, & 1.0 \end{bmatrix}^\top \tag{B.4}$$

Glider

The external forces are computed with a first-order Taylor expansion of the following longitudinal and lateral components:

$$f_{\text{lift}} = \frac{1}{2}\rho V_a^2 S\, C_L(\alpha, q, \delta_e) \qquad f_y = \frac{1}{2}\rho V_a^2 S\, C_Y(\beta, p, r, \delta_a, \delta_r) \tag{B.5}$$

$$f_{\text{drag}} = \frac{1}{2}\rho V_a^2 S\, C_D(\alpha, q, \delta_e) \qquad l = \frac{1}{2}\rho V_a^2 S\, bC_l(\beta, p, r, \delta_a, \delta_r) \tag{B.6}$$

$$m = \frac{1}{2}\rho V_a^2 S\, cC_m(\alpha, q, \delta_e) \qquad n = \frac{1}{2}\rho V_a^2 S\, bC_n(\beta, p, r, \delta_a, \delta_r) \tag{B.7}$$

where (i) f_{lift} and f_{drag} are rotated into f_x and f_y, (ii) S, c, b are the planar area of the wing surface, mean chord length, and wind span of the drone, (iii) α and β are the angle of attack and slip angle.

The state and action space are hypercubes with the following upper and lower limits:

$$X = \begin{bmatrix} -1000 & -1000 & -750 & -600 & -600 & -600 & -2 & -2 & -100 & -50 & -50 & -50 \\ 1000 & 1000 & -0.2 & 600 & 600 & 600 & 2 & 2 & 100 & 50 & 50 & 50 \end{bmatrix}^\top \tag{B.8}$$

$$U = \begin{bmatrix} -0.5 & -0.5 & -0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}^\top \tag{B.9}$$

Spacecraft

We model each spacecraft as a planar double integrator, with p_x and p_y positions in meters and v_x and v_y velocities in meters per second. We use a 1-dimensional finite element mesh for the net model, with four nodes.

The state and action spaces are hypercubes, where each x and y position is constrained to the hypercube $(p_x, p_y) \in [-3.0, 5.0]^2$, and velocities to $(v_x, v_y) \in [-0.25, 0.25]^2$.

We model the external forces on each spacecraft and net nodes to model the dynamics. The two controlled spacecraft are each actuated by forces in the x and y direction.

Between the net nodes and the edge net nodes and the controlled spacecraft, we consider a linear tension-only spring-damper. The symmetric force, aligned in the direction of the net, for net segment i is

$$F_{\text{net},i} = (l_i > l)(k_n(l_i - l) - c_n\dot{l}_i) \tag{B.10}$$

for nominal length l, net stiffness k_n, and damping coefficient c_n. Here the length l_i is the magnitude of the position difference between two nodes: $\|[p_{x,i}\ p_{y,i}]^\top - [p_{x,i-1}\ p_{y,i-1}]^\top\|$.

Between each of the net nodes and the target spacecraft, we model the collision as a stiff spring-damper, with a similar form to above. For node i, the contact force between the net node and the target is

$$F_{\text{contact},i} = (d_i < r_t)(k_c(d_i - r_t) - c_c\dot{d}_i) \tag{B.11}$$

for target radius r_t, collision stiffness k_c, and collision damping c_c. Here d_i is the distance between node i and the target: $\|[p_{x,i}\ p_{y,i}]^\top - [p_{x,t}\ p_{y,t}]^\top\|$. A visualization of the forces involved in this scenario is available in the figure below.

The reward is a composite function of three terms:

$$\begin{aligned} R(x) =& c_1 s(\|[p_{x,\text{centroid}}\ p_{y,\text{centroid}}]^\top - [p_{x,t}\ p_{y,t}]^\top\|, a_1) && \text{(B.12)}\\ &+ c_2 s(\|[v_{x,\text{centroid}}\ v_{y,\text{centroid}}]^\top - [v_{x,d}\ v_{y,d}]^\top\|, a_2) && \text{(B.13)}\\ &+ c_3 s(\|[v_{x,t}\ v_{y,t}]^\top - [v_{x,d}\ v_{y,d}]^\top\|, a_3) && \text{(B.14)}\\ D(x) \equiv& 0 && \text{(B.15)} \end{aligned}$$

where $[p_{x,\text{centroid}}\ p_{y,\text{centroid}}]^\top$ is the centroid of the controlled spacecraft and net structure. Here s is a normalization function: $s(d, a) = 1 - \frac{2}{\pi}\arctan(\frac{d}{a})$. The first term

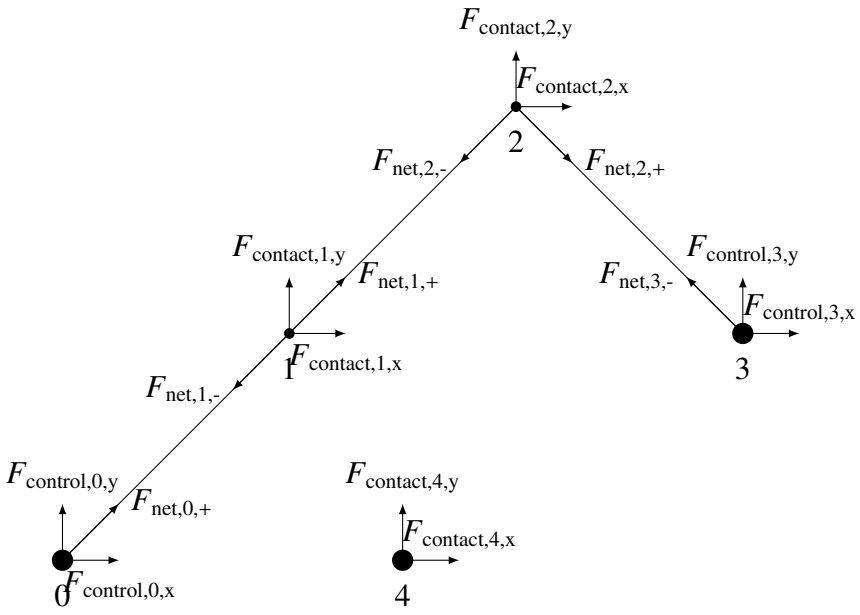


Figure B.1: Forces on Spacecraft Capture Problem for two net nodes.

serves to guide the centroid of the controlled spacecraft and net toward the target. The second and third terms encourage the captured target and controller spacecraft to move in the desired direction.

Appendix C

HYBRID TEMPORAL DIFFERENCE LEARNING FOR ADAPTIVE URBAN TAXI DISPATCH

This chapter is based on the publication:

> Benjamin Rivière and Soon-Jo Chung. "H-TD2: Hybrid Temporal Difference Learning for Adaptive Urban Taxi Dispatch". In: *IEEE Transactions on Intelligent Transportation Systems* (2021), pp. 1–10. DOI: 10.1109/TITS.2021.3097297.

C.1 Motivation

Coordinating a large fleet of automated taxis in complex and dynamic urban environments is an anticipated challenge for transportation network companies such as Uber, Lyft, Waymo, and Tesla. A typical urban mobility problem for these companies is taxi dispatch, where a fleet of taxis service customers and the remaining, idle taxis are coordinated with a dispatch algorithm to minimize the customer waiting time of future requests. In practice, a transportation network company might be composed of a dispatch center equipped with complete information and a large computational budget and a fleet of taxis, each operating with local information and a limited amount of processing power and communication bandwidth (see Fig. C.1). In this manner, the transportation network company network can be decomposed into an underlying star-topology network between taxis and the dispatch center, and an arbitrary peer-to-peer network between taxis. The proposed algorithm, H-TD^2, exploits this topology explicitly by proposing a hybrid algorithm with two distinct behaviors: the central node computes exact, large-batch policies infrequently, and each taxi computes approximate, online updates with local information.

The overview of H-TD^2 is shown in Fig. C.2. At a given timestep, the closest taxis service the new customer requests, and the rest of the free taxis are dispatched to reduce expected waiting time of future requests. The free taxis coordinate with a distributed game theoretic scheme to optimize their policy estimate, where the policy is estimated as follows: the servicing taxis communicate the customer data to their neighboring taxis, where the expected reward (e.g. customer waiting time) is estimated with a distributed estimation algorithm. Then, all taxis use the estimated

Figure C.1: Concept graphic of an intelligent transportation network. Autonomous taxis, that can include both ground and air vehicles, estimate in real-time the customer demand and coordinate locally to behave with bounded sub-optimality.

reward to update their policy estimate with temporal difference learning. If any of the taxis determine that their policy estimate error is larger than the user specified threshold, the taxi signals to the dispatch center for a centralized policy update.

The contributions of the paper are stated as follows:

> We derive a novel optimality bound by leveraging distributed estimation methods in local online policy estimation and introduce a trigger condition to the batch update, permitting the user to explicitly specify the policy's computational and communication expense vs. bounded sub-optimality trade-off. This advances the state of the art by enabling distributed operation with bounded sub-optimality.
>
> We propose a taxi-dispatch solution that is adaptive, model-free, and coordinated. Unlike state-of-the-art reinforcement-learning dispatch methods, our method directly adapts the policy based on real-time data, thereby providing a property of robustness to irregular urban mobility events such as traffic, weather, and major public events. This advancement is enabled by two step approach: first we propose a hybrid policy estimation in a finite-dimensional, agent-agnostic cell abstraction, and then we interface the resulting policy esti-

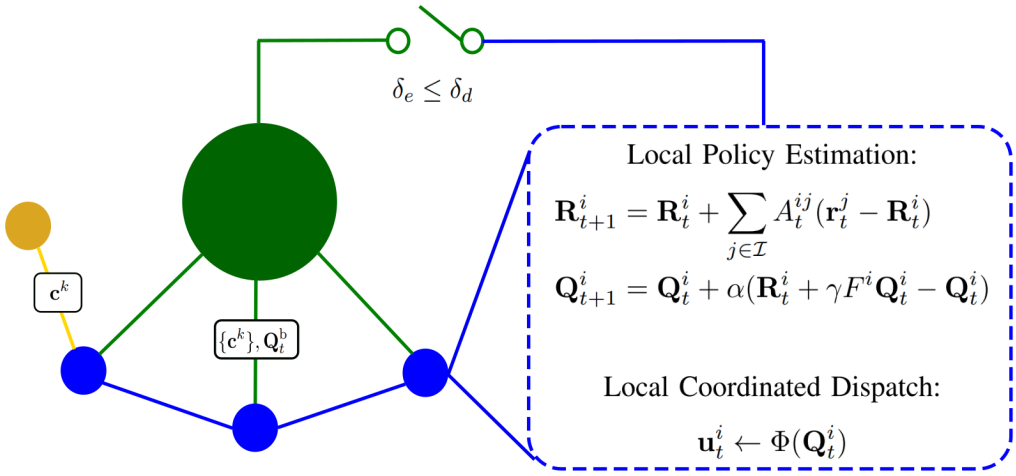


Figure C.2: Overview of H-TD^2.

mation for agent-based coordination with a local prescriptive game-theoretic task assignment.

We demonstrate the performance and computational properties of our method with numerical experiments: our algorithm reduces customer waiting time compared to a receding horizon control baseline and the simulation runtime is linear with the number of agents. We also validate our claim that adaptive algorithms are robust to general irregular events with a case study of the Chicago City taxis during the 2016 Major League Baseball World Series.

In Fig. C.2, blue represents the taxi network, yellow represents the customers, and green represents the dispatch center. The i$^{\text{th}}$ taxi estimates the dispatch policy with local operations: distributed estimation of reward, $\mathbf{R}^i_t$ computed in (C.9), and temporal difference learning to update the policy, $\mathbf{Q}^i_t$, (C.13). If any of the taxis determines that its policy estimate error, δ_e, is larger than the user specified threshold, δ_d, the taxi signals to the dispatch center to receive a centralized policy update, $\mathbf{Q}^{\mathrm{b}}_t$ (C.6). Finally, each free taxi uses the policy in a game theoretic formulation, Φ (C.20), to find its dispatch position vector, $\mathbf{u}^i_t$.

The remainder of the paper is organized as follows: in Sec. C.6, we review the related literature and compare our method with the state of the art. In Sec. C.2, we present the taxi dispatch problem description and a motivating example. In Sec. C.3, we present the cell-based Markov Decision Process (MDP). In Sec. C.4,

we discuss the exact and approximate solutions to the MDP and the integration of the learned policy into a game theoretic method. In Sec. C.5, we present numerical experiments demonstrating the advantages of our algorithm compared to a receding horizon control baseline in simulated and real customer datasets.

C.2 Problem Description

Notation: We denote vectors with a bold symbol, matrices with plain uppercase, scalars parameters with plain lowercase, functions with italics, and we use calligraphic symbols for operators and sets. We denote a taxi index with an i or j superscript, a customer index with a k superscript, and the time index with a subscript t. Also, I_n denotes the n-dimensional identity matrix.

Problem Statement: We consider the urban taxi dispatch problem, where we control a fleet of taxis to minimize customer waiting time. At each timestep, each customer requests is serviced by the nearest taxi. These *servicing* taxis use customer information to update their reward model and exchange information with neighboring taxis. The remaining *free* taxis are dispatched to locations in the map according to the proposed dispatch algorithm. The overall fleet control is summarized in Algorithm 5.

Algorithm 5: Fleet Control Problem

```
initialize taxi fleet;
for t ∈ [t_0 : t_f] do
    broadcast local customer requests to taxis;
    assign closest free taxis to service customers;
    dispatch free taxis to locations in the map;
end
```

The system, as shown in Fig. C.3, is composed of customers and taxis. The k^{th}-customer state, $\mathbf{c}^k$, is composed of the time of request, trip duration, pickup location, and dropoff location, i.e. $\mathbf{c}^k = [t_r^k, t_d^k, \mathbf{p}^{k,p}, \mathbf{p}^{k,d}]$ and its pickup location is shown in green in the top subplot. The i^{th}-taxi is defined by a position vector, $\mathbf{p}_t^i$ and an operation mode: free (shown in blue) or servicing (shown in orange). The dispatch solution is a desired position vector for each of the free taxis, $\mathbf{u}_t^i$, that results in minimizing customer waiting time over a time horizon. Our method estimates the optimal policy, visualized with the value function over the state-space in the bottom subplot, that maximizes the expected reward over time.

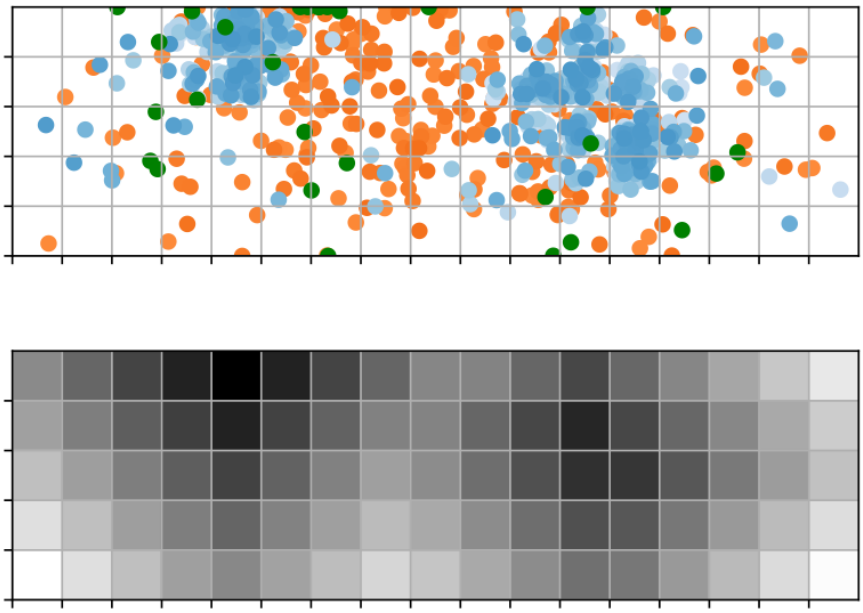


Figure C.3: State space representation of a Gridworld simulation with 1000 taxis, with corresponding value function estimation. In the top subplot, the blue dots are free taxis positions, the orange dots are positions of taxis currently servicing customers, and the green dots are the new customers requests pickup positions. In the bottom subplot, the approximate value function distribution is shown over the state space.

C.3 Cell-Based Markov Decision Process

The previous section described the dispatch problem with a agent-based perspective, i.e. in terms of positions and actions of individual taxis and customers. Next, we will introduce the cell-based Markov Decision Process (MDP), where cell-based refers to an Eulerian perspective in which we analyze values like location and reward with respect to cells of a discretized map as shown in Fig. C.3. The cell-based formulation decouples the decision making problem from the number of agents, permitting a finite dimensional policy representation. The decision making problem is formalized with a MDP, $\mathcal{M}$, defined as a tuple of state space, action space, transition model, reward model, and discount factor [2]:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle. \tag{C.1}$$

- The state space, $\mathcal{S}$, is defined as the set of map cells shown in Fig. C.3, where the state of the i^{th}-taxi, s_t^i is the cell index that contains that taxi's position. The number of cells in the environment is denoted by the cardinality of the set, $|\mathcal{S}|$.

- The action space, $\mathcal{A}$, is defined as a movement between map cells for for taxi i. The taxi on dispatch has 5 actions: $a_t^i \in \mathcal{A} = \{\text{stay}, \text{right}, \text{up}, \text{left}, \text{down}\}$, defined with respect to its current map cell in $\mathcal{S}$.

- The transition function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{P}$ is defined as $P(s_t^i, a_t^i, s_{t+1}^i) = \mathbb{P}(s_{t+1}^i | s_t^i, a_t^i)$. We define P with a deterministic, cell-based dynamical model, f:

$$s_{t+1}^i = f(s_t^i, a_t^i) \text{ where } P(s_t^i, a_t^i, s_{t+1}^i) = 1. \tag{C.2}$$

  If the next state is valid, the cell-based dynamical model moves the taxi from the initial state, s_t^i to the neighboring state s_{t+1}^i according to its action. If the next state is not valid, the dynamical model returns the initial state.

- The reward function $R_t : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined to be the negative of the expected customer waiting time and is estimated from reward samples; reward is the negative of the time it takes the taxi to go from its current position to the dispatch position defined by the cell-action, and then from that position to the customer request. The reward has a subscript t because we assume the reward changes over time due to the changing customer distribution. We equivalently write the R_t function as a vector of state-action pairs, $R_t(s, a) = \mathbf{R}_t[s|\mathcal{A}| + i(a)]$ and $\mathbf{R}_t \in \mathbb{R}^{n_q}$, where $n_q = |\mathcal{S}||\mathcal{A}|$ and $i(a)$ denotes the action's index. Given a customer request, $\mathbf{c}^k$, we compute a sample of the reward function, r_t^i that can be used to estimate the underlying reward, $\mathbf{R}_t$ according to an observation model:

$$r_t^i(s_t^i, a_t^i) = -(\eta(\mathbf{p}_t^i, \mathbf{u}_t^i) + \eta(\mathbf{u}_t^i, \mathbf{p}^{k,p})) \tag{C.3}$$

$$\mathbf{r}_t^i = H_t^i \mathbf{R}_t + \mathbf{v}_t^i \tag{C.4}$$

  Recall that $\mathbf{p}_t^i$ is the position of the i^{th}-taxi, $\mathbf{u}_t^i$ is the dispatch desired position, and $\mathbf{p}^{k,p}$ is the k^{th}-customer pickup position. Also, η is the estimated time-of-arrival function that accepts position vectors and returns a scalar time value. It is parameterized by the average taxi velocity $\overline{v}_{\text{taxi}}$. The measurement noise is sampled from a normal distribution with variance ς, $\mathbf{v}_t^i \sim \mathcal{N}(0, \varsigma I)$. Finally, the cell-based observation model, $H_t^i \in \mathbb{R}^{n_q \times n_q}$, is a binary diagonal matrix with unity elements at the state-actions pairs where the customer request $\mathbf{c}^k$ contains information of the corresponding state-action pair, and 0 otherwise.

- Note that γ is the discount rate of the system. This parameter determines the trade-off between greedy and long-term optimal behavior.

C.4 Algorithm Description and Analysis: H-TD^2

We describe the details of H-TD^2 in this section, defining the exact and approximate policy estimation, the hybrid switching behavior, and the game theoretic task assignment. The overview of the method is given in Algorithm 6.

Algorithm 6: H-TD^2 at timestep t

input: set of total, free, and servicing taxis: $\mathcal{I}, \mathcal{I}_f, \mathcal{I}_s$
output: action profile for free taxis, $\mathbf{u}_t$
`/* Hybrid Temporal Difference */`
for $\forall i \in \mathcal{I}$ **do**
  if $\delta_e > \delta_d$ (C.15) **then**
    slow update $\mathbf{Q}_t^i$ with aggregated global information (C.6) at the central node;
  else
    fast update $\mathbf{Q}_t^i$ with local information (C.13) at each taxi;
  end
end
`/* Game Theoretic Task Assignment */`
randomly initialize cell-based action profile $\mathbb{A}_t$;
while $\mathbb{A}_t$ *not converged* **do**
  randomly pick $i \in \mathcal{I}_f$ that has not converged;
  consider current action, a_t^i;
  propose random action $a_t^{i'}$;
  compute marginal utility, J with $\mathbf{Q}_t^i$ (C.20);
  stochastically assign action with J (C.21);
  check i^{th}-taxi action convergence;
end
Convert cell-based actions a_t^i to position vectors $\mathbf{u}_t^i$;

Centralized Q-value Computation

We present the idealized Bellman solution to the cell-based decision making problem specified in (C.1). The solution is a policy function that maps states to an action that maximizes the discounted reward over time and can be represented as a value function, as shown in Fig. C.3, or an action-value function known as Q_t^{b}-values. We use the latter and use the superscript b notation to denote the policy that is synthesized with a Bellman iteration method. We adopt the conventional optimal Q_t^{b}-value function as follows:

$$Q_t^{\text{b}}(s_t, a_t) = \mathbb{E}_{s\sim P(s_t'|s_t,a_t)}[R_t + \gamma \mathbb{E}_{a_t'\sim\pi^{\text{b}}} Q^{\text{b}}(s_t', a_t')] \tag{C.5}$$

We specify this general formulation with some assumptions. First, we use a finite-dimensional tabular Q_t^{b} and write the Q_t^{b} function as a vector of state-action pairs, $\mathbf{Q}_t^{\mathrm{b}} \in \mathbb{R}^{n_q}$, as done with the reward in Sec. C.3. Next, we apply the deterministic transition function, $P(s_t^i, a_t^i, s_{t+1}^i)$, as specified in (C.2) to remove the outer expectation. We remove the inner expectation by specifying the policy π^{b} to be a transition kernel matrix F^{b} such that $F^{\mathrm{b}}\mathbf{Q}_t^{\mathrm{b}}(s_t^i) = \max_{a_t^i} Q_t^{\mathrm{b}}(s_t^i, a_t^i)$. Combining these, we rewrite a simplified expression for $\mathbf{Q}_t^{\mathrm{b}}$ as the fixed point of the Bellman operator $\mathcal{T}$:

$$\mathbf{Q}_t^{\mathrm{b}} = \mathbf{R}_t + \gamma F^{\mathrm{b}}\mathbf{Q}_t^{\mathrm{b}} = \mathcal{T}\mathbf{Q}_t^{\mathrm{b}} \tag{C.6}$$

The Bellman iteration can be solved using conventional value iteration or policy iteration methods from batch customer data. For the purpose of this paper, we use a Modified Policy Iteration (MPI) method [3] to solve line 5 of Algorithm 6. However, there are complications with implementing a pure Bellman approach, which we address in the next subsection.

Distributed Reward Estimation and Q-value Iteration

We present a policy approximation that overcomes the Bellman solution's practical limitations. The first issue with a pure Bellman solution is that in an online setting, the reward information is hidden a priori and received incrementally in samples, $\mathbf{r}_t^i$. So the expectation of the reward is not immediately available. Furthermore, each taxi only has access to local information. To overcome these problems, we assume the hidden reward evolves as a random walk process and synthesize linear estimators for the hidden reward $\mathbf{R}_t$:

$$\mathbf{R}_{t+1} = \mathbf{R}_t + \mathbf{w}_t \tag{C.7}$$

$$\mathbf{R}_{t+1}^{\mathrm{c}} = \mathbf{R}_t^{\mathrm{c}} + \sum_{j \in \mathcal{I}} K_t^j(\mathbf{r}_t^j - H_t^j\mathbf{R}_t^{\mathrm{c}}) \tag{C.8}$$

$$\mathbf{R}_{t+1}^i = \mathbf{R}_t^i + \sum_{j \in \mathcal{I}} A_t^{ij}(\mathbf{r}_t^j - \mathbf{R}_t^i) \tag{C.9}$$

Recall $\mathbf{r}_t^i$ is the reward sample defined in (C.4). Also $\mathbf{R}_t^{\mathrm{c}}, \mathbf{R}_t^i \in \mathbb{R}^{n_q}$ are centralized and distributed reward estimators that will be used in the upcoming temporal difference learning, where the superscript c denotes a centralized quantity. The H_t^i matrix is the i^{th}-taxi's measurement model defined in (C.4) and the K_t^i matrix are the corresponding estimator gains. The process noise, $\mathbf{w}_t \sim \mathcal{N}(0, \varepsilon I)$ is sampled from a normal distribution where the parameter ε is computed offline from training data. The row-stochastic adjacency block matrix, A_t, specifies the local information

available to each agent and is defined as:

$$A_t^{ij} = B_t^{ij} / \sum_{j=1}^{n_i} B_t^{ij}, \text{ where } B_t^{ij} = \begin{cases} K_t^j H_t^j & j \in \mathcal{I}_t^i \\ \nvDash_{n_q \times n_q} & \text{else} \end{cases} \tag{C.10}$$

$$\mathcal{I}_t^i = \{ j \in \mathcal{I} \mid \|\mathbf{p}_t^i - \mathbf{p}_t^j\| < R_{\text{comm}} \}. \tag{C.11}$$

with diagonal block matrices $A_t^{ij}, B_t^{ij} \in \mathbb{R}^{n_q \times n_q}$ and full matrices $A_t, B_t \in \mathbb{R}^{n_i n_q \times n_i n_q}$. The i^{th} agent constructs the B_t^{ij} matrices from the gain and measurement matrices shared by its neighbors $j \in \mathcal{I}_t^i$. The local observation is parameterized by the radius of communication, R_{comm}.

The second issue with the Bellman approach is an intrinsic drawback that at higher state/action dimensions the Bellman-iteration calculation becomes computationally-expensive and cannot be quickly evaluated online. Instead, temporal difference learning [4] can be used as an approximate method to estimate $\mathbf{Q}$-values online using $\mathbf{R}_t^{\text{c}}$:

$$\mathbf{Q}_{t+1}^{\text{c}} = \mathbf{Q}_t^{\text{c}} + \alpha(\mathbf{R}_t^{\text{c}} + \gamma F^{\text{c}} \mathbf{Q}_t^{\text{c}} - \mathbf{Q}_t^{\text{c}}) \tag{C.12}$$

where $\alpha \in \mathbb{R}$ is the system learning rate and F^{c} is the transition kernel for this policy.

This formulation requires a central node to collect the data, compute a policy, and broadcast the new information at every timestep, scaling the computation complexity, bandwidth, and network delay with the number of taxis. To address this limitation, we introduce a distributed algorithm using communication between the taxis, and propose a policy update computed at each taxi using only local information:

$$\mathbf{Q}_{t+1}^{i} = \mathbf{Q}_t^{i} + \alpha(\mathbf{R}_t^{i} + \gamma F^{i} \mathbf{Q}_t^{i} - \mathbf{Q}_t^{i}) \tag{C.13}$$

where this temporal difference (C.13), with the distributed reward estimation (C.9) defines line 7 of Algorithm 6.

Using the approximate temporal difference method and estimating with only local information hurts the quality of the final policy used by each taxi. We derive the upper bound of the negative effect of these approximations through a optimality bound analysis, comparing the policy synthesized with the proposed algorithm (C.13) and the Bellman-optimal solution (C.6).

Theorem 15 *The expected distance between an arbitrary taxi* $\mathbf{Q}$*-value estimates,* $\mathbf{Q}_t^i$*, and the Bellman-optimal solution* $\mathbf{Q}_t^{\mathrm{b}}$ *is upper bounded by:*

$$\mathbb{E}\|\mathbf{Q}_t^i - \mathbf{Q}_t^{\mathrm{b}}\|_2 \leq \frac{2\sqrt{n_q(\varepsilon+\varsigma)}}{(1-\gamma)(1-\sqrt{1-\lambda_{\min}(\sum_{j\in\mathcal{I}} A_t^{ij})})} \tag{C.14}$$

where $\lambda_{\min}(.)$ *denotes the smallest eigenvalue of a matrix.*

Proof 14 *First, we write the distributed iteration as an application of the Bellman operator on the previous timestep with a disturbance. Then we solve for the disturbance to derive the final bound.*

Step 1*: Consider* (C.13) *and add and subtract* $\alpha\mathbf{R}_t$*:*

$$\begin{aligned}\mathbf{Q}_{t+1}^i &= \mathbf{Q}_t^i + \alpha(\mathbf{R}_t^i + \gamma F^i \mathbf{Q}_t^i - \mathbf{Q}_t^i) + \alpha\mathbf{R}_t - \alpha\mathbf{R}_t \\ &= \mathbf{Q}_t^i + \alpha(\mathbf{R}_t + \gamma F^i \mathbf{Q}_t^i - \mathbf{Q}_t^i) + \alpha(\mathbf{R}_t^i - \mathbf{R}_t) \\ &= \mathbf{Q}_t^i + \alpha(\mathcal{T}\mathbf{Q}_t^i - \mathbf{Q}_t^i) + \alpha\mathbf{e}_t^i\end{aligned}$$

where $\mathbf{e}_t^i = \mathbf{R}_t^i - \mathbf{R}_t$*. The system is contracting at rate* $1-\alpha(1-\gamma)$*, implying the system geometrically converges to an equilibrium about* $\mathcal{T}\mathbf{Q}_t^i = \mathbf{Q}_t^i$*. In addition, from Banach's fixed point theorem [5],* $\mathcal{T}$ *contracts to a unique fixed point,* $\mathbf{Q}_t^{\mathrm{b}}$*. Applying Discrete Gronwall's lemma [5]:*

$$\|\mathbf{Q}_t^i - \mathbf{Q}_t^{\mathrm{b}}\| \leq \frac{\|\mathbf{e}_t^i\|}{1-\gamma}$$

Note that the decaying initial condition term does not appear because we assume that the policy estimate is initialized with the Bellman solution, i.e. $\mathbf{Q}_0^i = \mathbf{Q}_0^{\mathrm{b}}$*.*

Step 2*: Here we need to bound the value* $\|\mathbf{e}_t^i\|$*, i.e. the error between the estimated reward and the true reward. We write the dynamics of the error vector* $\mathbf{e}_t^i$ *by subtracting* (C.7) *from* (C.9)*:*

$$\mathbf{e}_{t+1}^i = (I - \sum_{j\in\mathcal{I}} A_t^{ij})\mathbf{e}_t^i + \mathbf{d}_t$$

where $\mathbf{d}_t = \mathbf{w}_t + \sum_{j\in\mathcal{I}} A_t^{ij}\mathbf{v}_t^j$*. By applying Weyl's interlacing eigenvalue theorem [6], we prove that the* i^{th} *system is contracting at rate lower bounded by* $\lambda_t^i = 1 - \lambda_{\min}(\sum_{j\in\mathcal{I}} A_t^{ij})$*.*

We rewrite the disturbance as the product of an input matrix, M*, and the stacked noise,* $\mathbf{z}_t \sim \mathcal{N}(0, W)$*:*

$$\mathbf{d}_t = M_t\mathbf{z}_t$$

where $\mathbf{z}_t = \left[\mathbf{w}_t; \mathbf{v}_t^1; \ldots; \mathbf{v}_t^{n_i}\right]$, $M_t = \left[I_{n_q}, A_t^{i,1}, \ldots, A_t^{i,n_i}\right]$ *and* $W = \text{blkdiag}(\varepsilon I_{n_q}, \varsigma I_{n_q}, ..., \varsigma I_{n_q})$.

By application of the convergence theorem of discrete stochastic contracting systems [7, 8], the expected error of a single agent is upper bounded by:

$$\mathbb{E}\|\mathbf{e}_t^i\| \leq \frac{2\sqrt{C}}{1 - \sqrt{1 - \lambda_{\min}(\sum_{j\in\mathcal{I}} A_t^{ij})}}$$
$$C = \text{trace}(M_t^T M_t W)$$

It remains to calculate the value C:

$$C = \varepsilon \,\text{trace}(I_{n_q}) + \varsigma \sum_{j\in\mathcal{I}} \text{trace}\left((A_t^{ij})^T A_t^{ij}\right) \leq n_q(\varepsilon + \varsigma)$$

where we use the linearity of the trace operation to move it outside of the sum, then we use the non-negativity and row-stochasticity of A_t *to bound* $\sum_{j\in\mathcal{I}}(A_t^{ij})^T A_t^{ij} \leq I_{n_q}$. *The final result is found by plugging the result from Step 2 into the result from Step 1.*

Remark 9 *The optimality bound is driven by the contraction rate of the system* λ_t^i, *a combined graph and observeability quantity. Intuitively, this corresponds to a non-zero value when taxi i and its neighbors taxis j can measure the entire state-action vector. We can also consider a batch measurement over a time interval,* n_T *and an average contraction rate,* $\overline{\lambda_t^i}$. *This time interval approach exists in the multi-agent adaptive control literature, where* $\overline{\lambda_t^i} > 0$ *is analogous to an excitation level in the Collective Persistency of Excitation condition [9].*

Remark 10 *The proposed online method is used to estimate the optimal policy in a dynamic environment, i.e. the reward model,* $\mathbf{R}_t$ *is time-dependent. In this case, the optimal policy is non-stationary, i.e.* $\mathbf{Q}_{t+1}^{\text{b}} \neq \mathbf{Q}_t^{\text{b}}$. *In order to guarantee the convergence of the TD-algorithm, we require that there exists a timescale separation between the convergence of the TD-algorithm and the dynamics of* $\mathbf{Q}^{\text{b}}$*:*

$$\|\mathbf{Q}_{t+1}^{\text{b}} - \mathbf{Q}_t^{\text{b}}\| \ll (1-\gamma)(1 - \sqrt{1 - \lambda_{\min}(\sum A_t^{ij})})$$

Remark 11 *The adjacency matrix* A_t *dictates that each agent takes a convex combination of the neighboring measurements. Further, the estimation gain matrices,* K_t^i *are chosen with a Distributed Kalman Information Filter, whose proof of optimality with respect to mean-squared-error can be found in [10, 11]. Thus, the agents weigh the neighboring measurements appropriately.*

Hybrid Temporal Difference Algorithm

We define the switching condition in line 4 of Algorithm 6 with two parameters, δ_e, the estimated error in the system, and δ_d, the user specified desired error in the system.

Proposition 2 *If we define:*

$$\delta_e = \frac{2\sqrt{n_q(\varepsilon+\varsigma)}}{(1-\gamma)(1-\sqrt{1-\lambda_{\min}(\sum_{j\in\mathcal{I}} A_t^{ij})})} \tag{C.15}$$

the expected policy sub-optimality will be bounded by δ_d.

Nominally, the system evolves with the distributed temporal difference method (C.13), computing δ_e at each timestep. Each agent is able to compute this value because ε, ς, and γ are known system parameters and each agent keeps track of its own $\lambda_{\min}(\sum_{j\in\mathcal{I}} A_t^{ij})$ values. Applying the result from Theorem 15, the expected policy suboptimality is identically δ_e, so, if δ_e exceeds the desired error, δ_d, the desired sub-optimality is violated. However, if this condition occurs at time t, the system resets all taxis with a central policy update (C.6), i.e. $\mathbf{Q}_t^i = \mathbf{Q}_t^{\mathrm{b}}$, $\forall i \in \mathcal{I}$. Therefore, the H-TD^2 algorithm maintains the distance between the estimated policy and a true optimal policy to user specification. In effect, δ_d controls the trade-off of computational expense to policy sub-optimality, where $\delta_d \to 0$ produces a solution with no regret but maximum computational effort and $\delta_d \to \infty$ produces a solution with potentially infinite regret with little computational effort.

Game Theoretic Task Assignment

In this section, we propose a game-theoretic task assignment to coordinate the taxis according to the $\mathbf{Q}_t^i$-values estimated in Sec. 19. The $\mathbf{Q}_t^i$ policy does not account for the actions of the other taxis and, without additional coordination, the taxis would behave greedily by all going to the highest value cell, increasing the overall customer waiting time. To avoid this behavior, we design a potential game and a local action profile iteration to maximize each agent's marginal utility. This is implemented in Algorithm 6, Lines 10-18.

First, we introduce a global action profile, $\mathbb{A}_t$ and a local action profile for the i^{th} taxi, $\mathbb{A}_t^i$:

$$\mathbb{A}_t = \{a_t^j \mid \forall j \in \mathcal{I}\}, \text{ and } \mathbb{A}_t^i = \{a_t^j \mid \forall j \in \mathcal{I}_t^i\} \tag{C.16}$$

where the j^{th}-neighbor taxi communicates its action, a_t^j, to the i^{th}-taxi, where a_t^j is defined in (C.1).

Next, we introduce the current global fleet distribution Ω_t, i.e. the number of taxis in each cell, as a function of the action profile:

$$\Omega_t(s, \mathbb{A}_t) = \frac{1}{n_i} \sum_{j \in \mathcal{I}} \mathbb{I}(s = f(s_t^j, a_t^j)) \tag{C.17}$$

where $\mathbb{I}$ denotes the indicator function and f is the dynamics model specified in (C.1). The local fleet distribution, Ω_t^i, is found with the same calculation but summing only over the neighboring agents, $j \in \mathcal{I}_t^i$. By defining the radius of communication as $R_{\text{comm}} = 3\text{ds}$ where ds is the length of a cell in the environment, we guarantee that the i^{th} taxi can always calculate the fleet distribution in neighboring cells, $\mathcal{S}^i$, within one action of the current cell of the i^{th} taxi.

Next, we describe the desired fleet distribution using the $\mathbf{Q}$-values as computed in (C.9). Consider the following Boltzmann exploration strategy [4] strategy to synthesize a desired distribution, Ω^*, from the $\mathbf{Q}$-value estimates:

$$\Omega^*(s, Q_t^i) = \frac{\exp\left(\beta \max_{a \in \mathcal{A}} Q_t^i(s, a)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\beta Q_t^i(s, a')\right)} \tag{C.18}$$

Recall that $\mathcal{A}$ are the local actions available to each taxi defined in (C.1) and $\beta \in \mathbb{R}$ is an exploration/exploitation design constant. For simplicity of notation, we have written the action-values in its functional form, Q_t^i.

The goal of the game theoretic task assignment is to find an action profile, $\mathbb{A}_t$ through local iteration methods that minimizes the distribution distance between the current distribution, Ω_t and the desired distribution Ω^*. We describe a potential and noncooperative game meaning that the taxis will try to converge to a Nash equilibrium with a high potential function value. The global and marginal potential functions, Φ and J are defined as follows:

$$\Phi(\mathbb{A}_t) = -\sum_{s \in \mathcal{S}} (\Omega^*(s, Q_t^i) - \Omega(s, \mathbb{A}_t))^2 \tag{C.19}$$

$$J(\mathbb{A}_t^i) = -\sum_{s \in \mathcal{S}^i} (\Omega^*(s, Q_t^i) - \Omega(s, \mathbb{A}_t^i))^2 \tag{C.20}$$

For the calculation of J, we only require the indices that correspond to neighboring cells of the current cell of the i^{th}-taxi, thereby permitting a local calculation.

Remark 12 *The game's utility function, $\Phi(\mathbb{A}_t)$ has an analogy to sample-based planners if each taxi in the fleet is considered as a sampled action of a stochastic policy, $\Omega^*(s, \mathbf{Q}_t^i)$. This choice of utility function is interesting because it could be the utility function chosen by a centralized algorithm but we can maximize it with local calculations through J.*

Remark 13 *Note that J is indeed the marginal contribution on the global potential function:*

$$\Phi(\mathbb{A}_t') - \Phi(\mathbb{A}_t) = J(\mathbb{A}_t^{i'}) - J(\mathbb{A}_t^i)$$

where $\mathbb{A}_t^{'} = \{a_t^j | \ \forall j \in \mathcal{I}/\{i\}\} \cup \{a_t^{i'}\}$ is the global alternative action set.

We use a game-theoretic reinforcement learning technique, binary log-linear learning [12] to iterate to an action set $\mathbb{A}_t$, shown in line 10-19 of Algorithm 6. At each timestep, t, the action set, $\mathbb{A}_t$ is randomly initialized. While all other taxi's actions are held, a randomly selected i^{th}-taxi chooses between the previously held action, a_t^i, and an alternate action $a_t^{i'}$ with probability $p_t^i(\mathbb{A}_t^i, \mathbb{A}_t^{i'})$:

$$p_t^i(\mathbb{A}_t^i, \mathbb{A}_t^{i'}) = \frac{\exp\left(J(\mathbb{A}_t^i)/\tau\right)}{\exp\left(J(\mathbb{A}_t^i)/\tau\right) + \exp\left(J(\mathbb{A}_t^{i'})/\tau\right)} \tag{C.21}$$

where $\mathbb{A}_t^{i'} = \{a_t^j | \ \forall j \in \mathcal{I}_t^i/\{i\}\} \cup \{a_t^{i'}\}$ is the local alternative action set. The coefficient $\tau \in \mathbb{R}_{>0}$ is a design parameter specifying how likely taxi i chooses a sub-optimal action, to specify the trade-off between exploration and exploitation. The action set is chosen once the iteration has converged, completing the game-theoretic task assignment. Then, the cell-based action a_t^i is converted to a dispatch vector $\mathbf{u}_t^i$, where $\mathbf{u}_t^i$ is a randomly sampled position vector in the cell after the dispatch action is taken.

C.5 Numerical Experiments

Baseline and Variants

We compare our H-TD^2 solution with a receding horizon control (RHC) baseline dispatch algorithm, adapted from the baseline in [13]. For this section, we use independent notation from the rest of the paper, matching the notation in [13]. The

RHC dispatch algorithm is formulated as the following linear program:

$$\begin{aligned} \mathbf{u}^* = \arg\max \sum_{t=t_0}^{t_0+t_{\text{rhc}}} \gamma^{t-t_0} \sum_{i=1}^{M} \min(\overline{w}_{t,i} - \mathbf{x}_{t,i}, 0) \quad \text{s.t.} \\ \sum_{j=1}^{M} u_{ij,t} = \mathbf{x}_{t+1,i}, \quad \sum_{i=1}^{M} u_{ij,t} = \mathbf{x}_{t,i} \\ u_{ij,t} = 0 \quad \forall j \notin \mathcal{S}^i, \quad \mathbf{x}_{t_0,i} = X_{0,i} \end{aligned} \tag{C.22}$$

where the state variable, $\mathbf{x}_t \in \mathbb{Z}^{|S|}$, is the number of free taxis in each cell, the control variable $u_{ij,t}$ is the number of taxis moving from cell i into cell j at time t and t_{rhc} is the RHC planning horizon. The first two constraints are conservation constraints: (i) the number of taxis in cell i is the number of taxis moving into cell i, and (ii) the number of taxis moving from cell i is equal to the number of taxis previously in cell i. The third constraint is that each taxi can only move to a neighboring cell. The fourth constraint is the initial condition. For large fleets, a proper assumption from [13] is to relax $\mathbf{u}_{ij,t}$ from integers to real numbers, resulting in a linear program in $\mathbf{u}_{ij,t}$. The reward is the difference of customer demand and taxi supply, where $\overline{w}_{t,i}$ is the expected customer demand and is computed from offline training data with a calculation proposed in [13, 14]. The baseline is chosen to demonstrate the advantage of fast adaption over learned prediction: static prediction methods (either explicit in model-based or implicit in model-free) are vulnerable to events occurring outside of the training domain.

We study the effect of each component of the policy estimation algorithm by comparing variants of the policy: *Centralized Temporal Difference* (C-TD), *Distributed Temporal Difference* (D-TD), and *Bellman-optimal*. All of these variants control the fleet with binary log-linear learning (C.21), but they differ in how $\mathbf{Q}$ is synthesized: C-TD uses (C.8) and (C.12), D-TD uses (C.9) and (C.13), and the Bellman-optimal policy is synthesized with (C.6). Equivalently, the D-TD and Bellman-optimal algorithms can be interpreted as the limiting behavior of H-TD^2 corresponding to the respective cases where $\delta_d = \infty$ and $\delta_d = 0$.

Customer Demand Datasets

We consider two datasets of customer requests: a synthetic dataset for a Gridworld environment and the real customer taxi dataset from the city of Chicago [15]. Recall each customer request is defined as follows: $\mathbf{c}^k = [t_r^k, t_d^k, \mathbf{p}^{k,p}, \mathbf{p}^{k,d}]$.

The synthetic dataset is generated as follows: At each timestep, $t \in [t_0, t_0 + \Delta_t]$, the customer request model is sampled n_c times, where n_c is the number of cus-

tomers per timestep of the simulation. The time of the request, t_r^k is uniformly randomly sampled in the timestep. The pickup location, $\mathbf{p}^{k,p}$, is found by sampling a 2-dimensional Gaussian Mixture Model (GMM), where translating Gaussian distributions capture the underlying dynamic customer demand. The dropoff location, $\mathbf{p}^{k,d}$ is a randomly sampled position in the map, and the duration of the trip, t_d^k is given by $\eta(\mathbf{p}^{k,p}, \mathbf{p}^{k,d})$. The GMM model is parameterized by the number of Gaussian distributions, n_G, the speed of the distributions, v_G, the variance, σ_G, and the initial position, and unit velocity vector of the centroid for each distribution.

The real customer dataset is taken from the city taxi dataset of Chicago [15] filtered by start and end timestamp. The Socrata API permits importing raw data in the $\mathbf{c}^k$ format, where the location data is specified in longitude, latitude coordinates. For this experiment, we load a city map of Chicago as a shapefile and perform minor geometric processing with the Shapely Python toolbox.

Results

Gridworld Simulations: We present variants of our H-TD^2 algorithm against a RHC baseline in a Gridworld environment as shown in Fig. C.3. This flexible, synthetic environment permits us to test on a range of system parameters.

First, we introduce the algorithms in a small-scale simulation. We synthesize a dataset with parameters: $n_c = 5$, $n_G = 2$, $v_G = 0.02625$, $\sigma_G = 0.014$, and randomly initialize the mean and direction of the distributions. Our simulation parameters are: $n_i = 100$, $|\mathcal{S}| = 85$, $\gamma = 0.9$, $\alpha = 0.75$, $n_T = 10$, $\varsigma = 0.014$, $\varepsilon = 0.0187$, $\delta_d = 0.025\|\mathbf{Q}_0^{\mathrm{b}}\|$, $\beta = 150$, $\overline{v}_{\mathrm{taxi}} = 0.125$, $\tau = 0.0001$, and $t_{\mathrm{RHC}} = 10$. We run this experiment for each of the algorithms for 5 trials. This experiment is shown in Fig. C.3, modified with $n_i = 1000$.

Next, we collect statistics on the cumulative reward of each algorithm, and plot the results in Fig. C.4. The algorithms behave as expected: in descending order of performance, Bellman, centralized temporal difference, H-TD^2, distributed temporal difference, followed by the receding horizon control baseline. At the cost of computational effort, the user can tune the performance of H-TD^2 between the distributed temporal difference and Bellman-optimal solution by changing δ_d. The simulation is run 5 times, and the mean with standard deviations is visualized in the plot.

To explain the performance difference between the variants of our method, we show an error trace of the policy in Fig. C.5. For a given policy $\mathbf{Q}$, we calculate the error

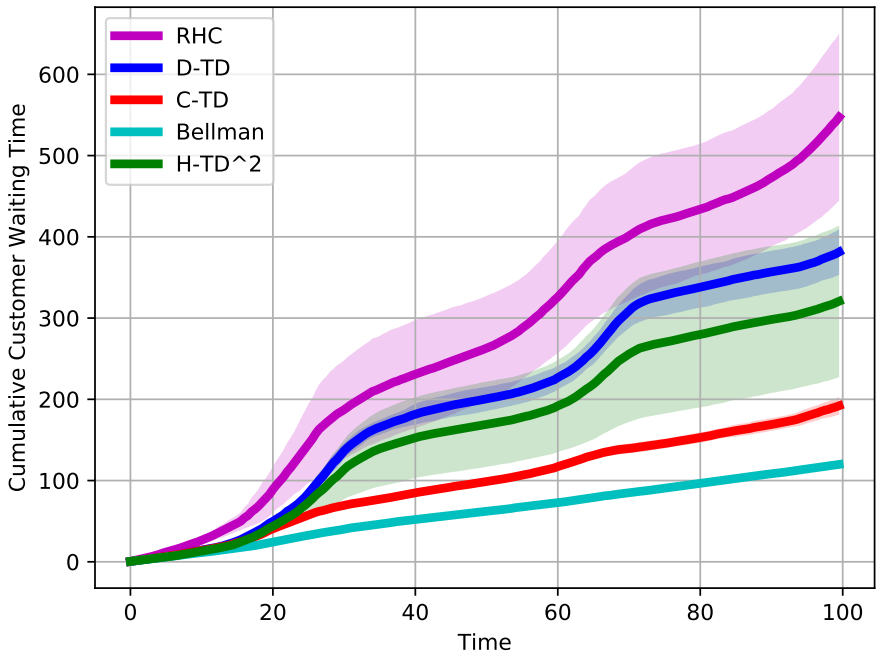


Figure C.4: Cumulative customer waiting time for different algorithms in the small-scale Gridworld environment. The algorithms behave as expected: in descending order of performance, Bellman, centralized temporal difference, H-TD^2, distributed temporal difference, followed by the receding horizon control baseline. The simulation is run 5 times, and the mean with standard deviations is visualized in the plot.

with respect to the Bellman solution: $e_t^{\mathbf{Q}} = \|\mathbf{Q}_t^{\mathrm{b}} - \mathbf{Q}_t\| / \|\mathbf{Q}_t^{\mathrm{b}}\|$. For this experiment, we set the δ_d parameter is set to 2.5 % of the norm of the Bellman solution, indicated by the dashed horizontal line. Initially, the H-TD^2 and distributed temporal difference (D-TD) algorithms behave identically, until the trigger condition is satisfied and the H-TD^2 requests a global Bellman-optimal update, thereby bringing its error to zero. As expected, the centralized-temporal difference method, C-TD, generally tends to estimate the $\mathbf{Q}$-values better than its distributed counterpart, D-TD.

Next, we test the proposed algorithm and baseline's scalability and performance across a wide range of taxi-densities and plot the results in Fig. C.6. We fix the parameters from the small-scale simulation and only change the number of taxis and number of customers, where we maintain the ratio $n_i/n_c = 10$. In the top figure, the average reward is shown across a variety of taxi density regimes, where the H-TD^2 algorithm outperforms a RHC baseline by almost a factor of 2 in all taxi-density regimes. In the bottom figure, we show that the computational time is approximately linear with number of taxis (and taxi-density) across 3 orders of

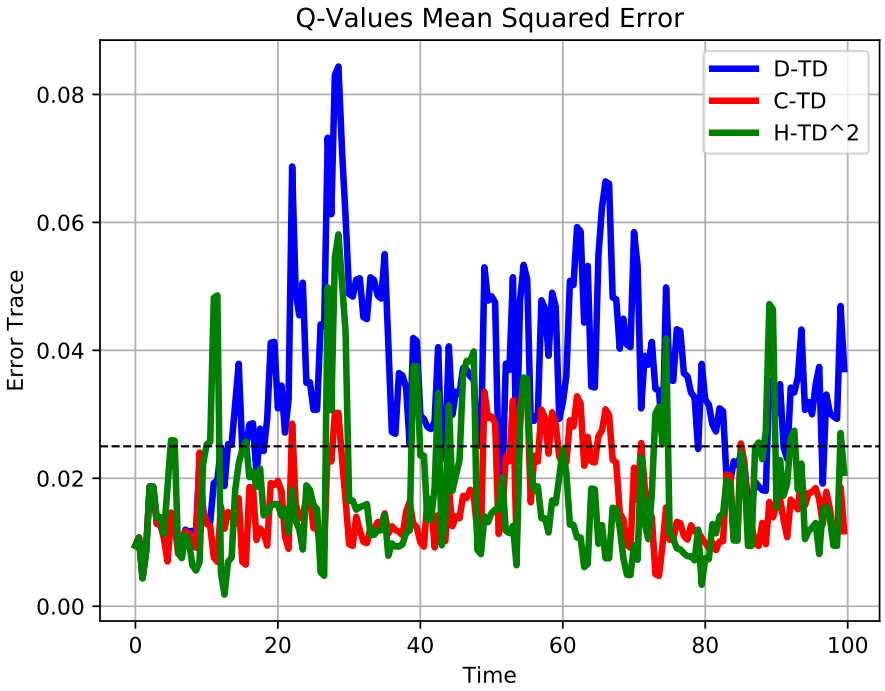


Figure C.5: **Q**-value error trace for different algorithms with respect to the Bellman-optimal. Initially, the H-TD^2 and distributed temporal difference algorithms behave identically, until the trigger condition is satisfied and the H-TD^2 requests a global Bellman-optimal update, bringing the error to zero. The δ_d parameter is set to 2.5 % of the norm of the Bellman solution and is shown with a dashed black horizontal line.

magnitude. The computational complexity of both RHC and H-TD^2 scales with the spatial resolution of the simulation, and in practice, we limit the maximum number of cells to 200.

Chicago City Simulations We present the H-TD^2 against an RHC baseline using real customer data from the city of Chicago public dataset [15], in a Chicago map environment. We show that our algorithm outperforms the baseline in practical datasets and demonstrate that online algorithms are robust in irregular urban mobility events.

In Fig. C.7, we present the Chicago city taxi customer demand across an irregular event: Game 5 of the 2016 Baseball World Series. The map cells show the number of customer pickup requests, and the green star is Wrigley Field's (baseball stadium) location. Below, we plot the customer demand over time for the cell containing Wrigley Field. We show that a reward model trained using data from the day before would not accurately predict the behavior of the next day.

We evaluate the algorithms and plot the results in Fig. C.8. Our simulation param-

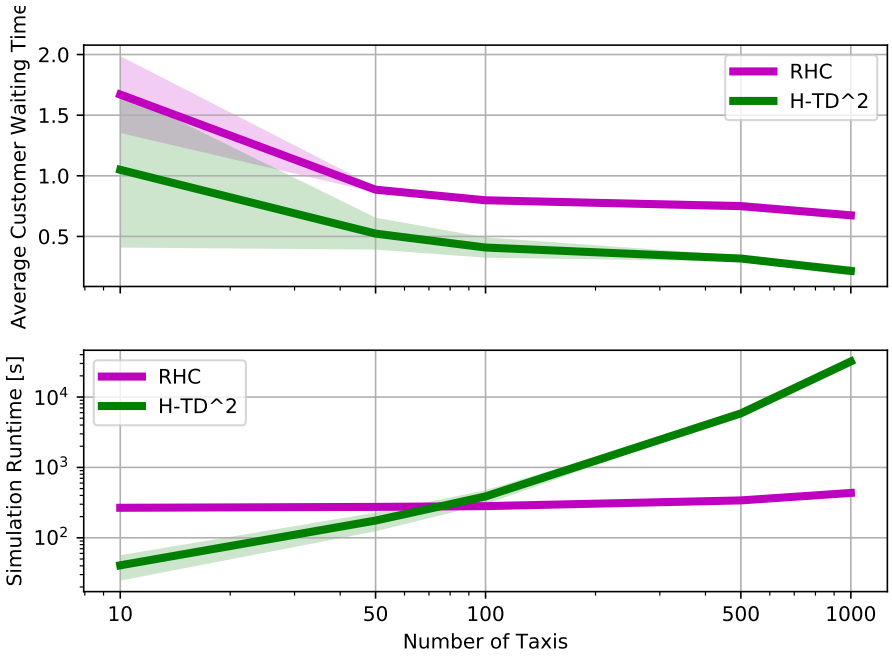


Figure C.6: Performance and scalability analysis of H-TD^2 and RHC against number of taxis. In the top subplot, the average reward is shown across a variety of taxi density regimes, and the proposed algorithm outperforms a receding horizon control baseline by at least 50 % in all taxi-density regimes. In the bottom subplot, the computational time is approximately linear with number of taxis (and taxi-density) across 3 orders of magnitude.

eters are: $n_i = 2000$, $|\mathcal{S}| = 156$, $\gamma = 0.8$, $\alpha = 0.1$, $n_T = 10$, $\varsigma = 0.0001$, $\varepsilon = 0.0001$, $\delta_d = 0.025\|\mathbf{Q}_0^{\text{b}}\|$, $\beta = 1$, $\overline{v}_{\text{taxi}} = 22$ miles per hour, $\tau = 0.0001$, and $t_{\text{RHC}} = 10$. We train a reward model using the data from October 29$^{\text{th}}$, 2016 with a total of 91, 165 customer requests. Then, we collect 54, 115 customer requests from October 30$^{\text{th}}$, 2016, which we reveal real-time to the H-TD^2 and RHC dispatch algorithms. In total, the H-TD^2 algorithm has a total customer waiting time of 501 hours an improvement of 26 % over the RHC baseline of a cumulative customer waiting time of 684 hours. This result demonstrates the robustness of adaptive algorithms to irregular events.

C.6 Related Work

Recent urban mobility research has developed dynamic and scalable methods. A well-studied example is the vehicle routing and dial-a-ride problems [16, 17] where taxis find a minimum cost path through a routing graph, and its dynamic extension, where part or all of the customer information is unknown and revealed dynamically. Recent dynamic routing research proposes scalable solutions to dynamic

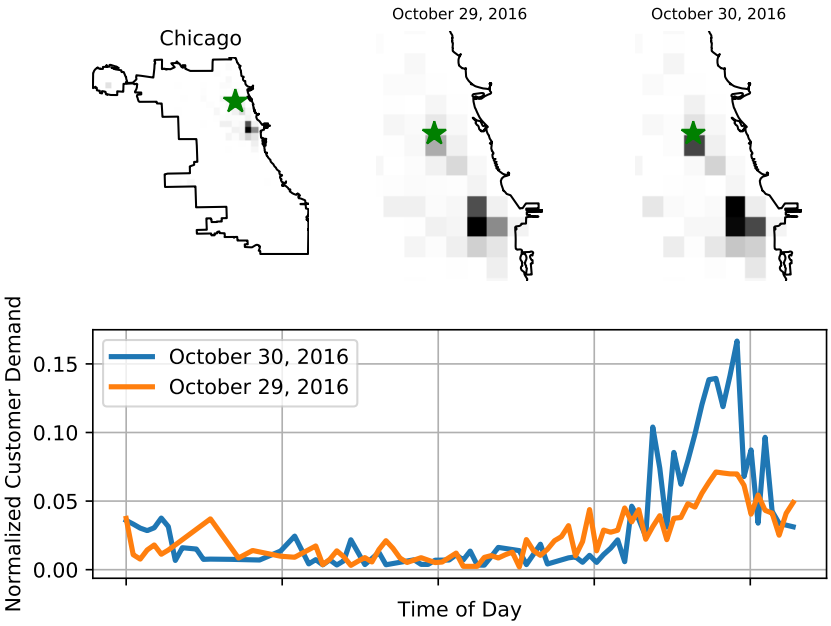


Figure C.7: Chicago city taxi customer demand across an irregular event: Game 5 of the 2016 Baseball World Series. The map cells show the number of customer pickup requests, and the green star is Wrigley Field's location. Below, we plot the customer demand over time for the cell containing Wrigley Field to show that a reward model trained using data from the day does not accurately predict the future behavior.

routing problems with bio-inspired methods [18], data-driven methods [19–21], and model-based methods [22, 23]. In this paper, we study a variant of the dynamic routing problem, taxi dispatch, where we propose a novel two-stage approach: distributed estimation with temporal difference learning, and game-theoretic coordination. This advances the state of the art by permitting adaptive distributed operation with bounded sub-optimality with respect to the optimal centralized policy.

Taxi dispatch is an emerging urban mobility problem where free taxis are dispatched to locations in the map to minimize customer waiting time of future requests. Recent approaches have adopted model-based [14, 24] and model-free [13] methods. An online model-based method like [14] uses real-time data to fit a system prediction model (for example, customer demand and taxi supply) and then compute a receding horizon control solution in response to that model. Pre-specified system models can be over-restrictive, and recent reinforcement-learning model-free methods [13, 25, 26] have been used successfully to overcome this limitation. In a model-free method, events are not explicitly modelled, they are captured by the arbitrary dynamics of the underlying reward. In regular operation, this reward is

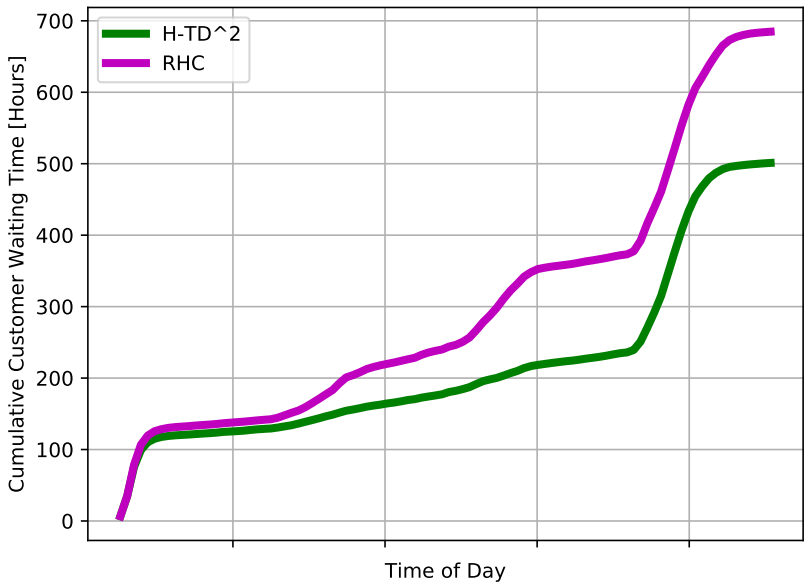


Figure C.8: Cumulative customer waiting time for the H-TD^2 and the RHC baseline in a Chicago city environment with with a fleet of 2,000 taxis servicing 54,115 real customer requests during the 2016 Major League Baseball World Series. The H-TD^2 algorithm has a total customer waiting time of 504 hours, an improvement of 26 % over the RHC baseline.

periodic, and can be accurately predicted (either explicitly or implicitly) and used for fleet control. However, it is possible that an irregular event occurs out of training domain and causes the reward dynamics to be unpredictable. In this case, we argue that it is better to adapt in real-time than predict with irrelevant data. Our model-free approach adapts the policy directly in response to real-time data, achieving performance that is robust to unpredictable, irregular events such as weather, accidents, and major public events.

Our method leverages results from reinforcement learning in convergence of temporal difference iteration [27–29] in a dynamic environment, i.e. when the reward or transition probabilities are changing over time. An alternative online model-free approach is online actor-critic [30], where our work differs from this result in two ways: we consider a general non-quadratic reward function and we consider a multi-agent setting by analyzing a hierarchical system of a temporal difference iteration with distributed estimation of the reward model. To the best of the authors knowledge, the only other work to propose a distributed temporal difference algorithm is recent work [31] that addresses the convergence properties of consensus on model parameters in the case of linear function approximations.

In general, multi-agent reinforcement learning research is challenging because the MDP's state and action space dimensionality is coupled to the number of agents, which is typically handled by using either (i) function approximation methods such as deep neural networks or (ii) decoupled, decentralized solutions. A survey paper on multi-agent reinforcement learning discusses additional methods [32]. In contrast to an agent-based (or Lagrangian) approach, our method uses a naturally scalable cell-based (or Eulerian) model that decouples the problem dimensionality from the number of agents, inspired by a method used in probabilistic swarm guidance [33].

Because of the cell-based abstraction, our algorithm requires an additional task assignment component to coordinate taxis. Task assignment is a canonical operations research problem and there exists many available centralized [34–37] and decentralized [38–40]. Among these options, we use a distributed prescriptive game theory [12] approach that leverages existing asymptotic game theoretic optimality and convergence results. In contrast to conventional *descriptive* game theory, *prescriptive* game theory designs multi-agent local interactions to achieve desirable global behavior. Using one such method, binary log-linear learning [12], the taxis achieve global cooperative behavior with only local information.

BIBLIOGRAPHY

[1] Benjamin Rivière and Soon-Jo Chung. "H-TD2: Hybrid Temporal Difference Learning for Adaptive Urban Taxi Dispatch". In: *IEEE Transactions on Intelligent Transportation Systems* (2021), pp. 1–10. DOI: 10.1109/TITS.2021.3097297.

[2] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

[3] Puterman and Shin. "Modified Policy Iteration Algorithms for Discounted Markov Decision Problems". In: *Management Science* 24.11 (1978).

[4] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

[5] A M. Stuart and Tony Humphries. *Dynamical Systems and Numerical Analysis*. 1996.

[6] Roger A. Horn and Charles R. Johnson. *Matrix Analysis, 2nd Ed*. Cambridge University Press, 2012.

[7] Winfried Lohmiller and Jean-Jacques E. Slotine. "On Contraction Analysis for Non-Linear Systems". In: *Automatica* 34.6 (1998), pp. 683–696. DOI: 10.1016/S0005-1098(98)00019-3.

[8] H. Tsukamoto and S. -J. Chung. "Robust Controller Design for Stochastic Nonlinear Systems via Convex Optimization". In: *IEEE Trans. Autom. Control* (2021), pp. 1–1. DOI: 10.1109/TAC.2020.3038402.

[9] Patrick M. Wensing and Jean-Jacques E. Slotine. "Cooperative Adaptive Control for Cloud-Based Robotics". In: *Proc. IEEE Int. Conf. Robot. Autom.* 2018, pp. 6401–6408.

[10] Reza Olfati-Saber. "Kalman-Consensus Filter : Optimality, stability, and performance". In: *Proc. IEEE Int. Conf. Decision Control*. 2009, pp. 7036–7042.

[11] Saptarshi Bandyopadhyay and Soon-Jo Chung. "Distributed Bayesian filtering using logarithmic opinion pool for dynamic sensor networks". In: *Automatica* 97 (2018), pp. 7–17.

[12] Jason R. Marden and Jeff S. Shamma. "Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation". In: *Games Econ. Behav.* 75.2 (2012), pp. 788–808.

[13] Takuma Oda and Carlee Joe-Wong. "MOVI: A Model-Free Approach to Dynamic Fleet Management". In: *INFOCOM*. IEEE, 2018, pp. 2708–2716.

[14] Fei Miao et al. “Taxi Dispatch With Real-Time Sensing Data in Metropolitan Areas: A Receding Horizon Control Approach”. In: *IEEE Trans. Autom. Sci. Eng.* 13.2 (2016), pp. 463–478.

[15] “Chicago Data Portal”. In: (). https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew.

[16] Sin C. Ho et al. “A survey of dial-a-ride problems: Literature review and recent developments”. In: *Transportation Research Part B: Methodological* 111 (2018), pp. 395–421. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2018.02.001.

[17] Victor Pillac et al. “A review of dynamic vehicle routing problems”. In: *Eur. J. Oper. Res.* 225.1 (2013), pp. 1–11.

[18] Rutger Claes, Tom Holvoet, and Danny Weyns. “A Decentralized Approach for Anticipatory Vehicle Routing Using Delegate Multiagent Systems”. In: *IEEE Trans. Intell. Transp. Syst.* 12.2 (2011), pp. 364–373.

[19] Kunal Menda et al. “Deep Reinforcement Learning for Event-Driven Multi-Agent Decision Processes”. In: *IEEE Trans. Intell. Transp. Syst.* 20.4 (2019), pp. 1259–1268.

[20] Abubakr O. Al-Abbasi, Arnob Ghosh, and Vaneet Aggarwal. “DeepPool: Distributed Model-Free Algorithm for Ride-Sharing Using Deep Reinforcement Learning”. In: *IEEE Trans. Intell. Transp. Syst.* 20.12 (2019), pp. 4714–4727.

[21] Jintao Ke et al. “Optimizing Online Matching for Ride-Sourcing Services with Multi-Agent Deep Reinforcement Learning”. In: *CoRR* abs/1902.06228 (2019).

[22] Renshi Luo, Ton J. J. van den Boom, and Bart De Schutter. “Multi-Agent Dynamic Routing of a Fleet of Cybercars”. In: *IEEE Trans. Intell. Transp. Syst.* 19.5 (2018), pp. 1340–1352.

[23] Zhao Zhou et al. “Two-Level Hierarchical Model-Based Predictive Control for Large-Scale Urban Traffic Networks”. In: *IEEE Trans. Contr. Sys. Techn.* 25.2 (2017), pp. 496–508.

[24] Rick Zhang and Marco Pavone. “Control of robotic mobility-on-demand systems: A queueing-theoretical perspective”. In: *Int. J. Robot. Res.* 35.1-3 (2016), pp. 186–203.

[25] Xiaocheng Tang et al. “A Deep Value-network Based Approach for Multi-Driver Order Dispatching”. In: *KDD*. ACM, 2019, pp. 1780–1790.

[26] Zhe Xu et al. “Large-Scale Order Dispatch in On-Demand Ride-Hailing Platforms: A Learning and Planning Approach”. In: *KDD*. ACM, 2018, pp. 905–913.

[27] Michael L. Littman and Csaba Szepesvári. “A Generalized Reinforcement-Learning Model: Convergence and Applications”. In: *Proc. Int. Conf. Machine Learn.* 1996.

[28] Balázs Csanád Csáji and László Monostori. “Value Function Based Reinforcement Learning in Changing Markovian Environments”. In: *J. Mach. Learn. Res.* 9 (2008), pp. 1679–1709.

[29] Csaba Szepesvári and Michael L. Littman. “A Unified Analysis of Value-Function-Based Reinforcement-Learning Algorithms”. In: *Neural Computation* 11.8 (1999), pp. 2017–2060.

[30] Kyriakos G. Vamvoudakis and Frank L. Lewis. “Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem”. In: *Autom.* 46.5 (2010), pp. 878–888.

[31] Thinh T. Doan, Siva Theja Maguluri, and Justin Romberg. “Finite-Time Analysis of Distributed TD(0) with Linear Function Approximation on Multi-Agent Reinforcement Learning”. In: *ICML*. Vol. 97. Proc. Machine Learn. Res. PMLR, 2019, pp. 1626–1635.

[32] Lucian Busoniu, Robert Babuska, and Bart De Schutter. “Multi-Agent Reinforcement Learning: A Survey”. In: *Proc. Int. Conf. Contr. Autom. Robot. Vision*. 2006, pp. 1–6.

[33] Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y. Hadaegh. “Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents”. In: *IEEE Trans. Robot.* 33.5 (2017), pp. 1103–1123.

[34] Harold W. Kuhn. “The Hungarian Method for the Assignment Problem”. In: *50 Years of Integer Programming*. Springer, 2010, pp. 29–47.

[35] D. P. Bertsekas. “The auction algorithm: A distributed relaxation method for the assignment problem”. In: *Annals of Operations Research* 14.1 (1988), pp. 105–123.

[36] Dimitri P. Bertsekas and David A. Castañon. “Parallel Synchronous and Asynchronous Implementations of the Auction Algorithm”. In: *Parallel Comput.* 17.6-7 (1991), pp. 707–732.

[37] John Bellingham et al. “Multi-Task Allocation and Path Planning for Cooperating UAVs”. In: *Cooperative Control: Models, Applications and Algorithms*. Ed. by Sergiy Butenko, Robert Murphey, and Panos M. Pardalos. Boston, MA: Springer US, 2003, pp. 23–41.

[38] Daniel Morgan et al. “Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming”. In: *Int. J. Robot. Res.* 35.10 (2016), pp. 1261–1285.

[39] D. Dionne and C. A. Rabbath. “Multi-UAV Decentralized Task Allocation with Intermittent Communications: the DTC algorithm”. In: *Proc. American Control Conf.* 2007, pp. 5406–5411.

[40] P. B. Sujit and R. Beard. “Distributed Sequential Auctions for Multiple UAV Task Allocation”. In: *2007 American Control Conf.* 2007, pp. 3955–3960.