Robust Distributed Planning Strategies for Autonomous Multi-Agent Teams

by

Sameera S. Ponda

S.M. in Aeronautics and Astronautics,
Massachusetts Institute of Technology (2008)
S.B. in Aerospace Engineering with Information Technology,
Massachusetts Institute of Technology (2004)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012

Author .....
Department of Aeronautics and Astronautics
August 23, 2012

Certified by .....
Jonathan P. How
Richard C. Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Mary L. Cummings
Associate Professor of Aeronautics and Astronautics

Certified by .....
Devavrat Shah
Jamieson Associate Professor of Electrical Engineering and Computer Science

Accepted by .....
Eytan H. Modiano
Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Robust Distributed Planning Strategies for Autonomous Multi-Agent Teams

by

Sameera S. Ponda

Submitted to the Department of Aeronautics and Astronautics
on August 23, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Aeronautics and Astronautics

Abstract

The increased use of autonomous robotic agents, such as unmanned aerial vehicles (UAVs) and ground rovers, for complex missions has motivated the development of autonomous task allocation and planning methods that ensure spatial and temporal coordination for teams of cooperating agents. The basic problem can be formulated as a combinatorial optimization (mixed-integer program) involving nonlinear and time-varying system dynamics. For most problems of interest, optimal solution methods are computationally intractable (NP-Hard), and centralized planning approaches, which usually require high bandwidth connections with a ground station (e.g. to transmit received sensor data, and to dispense agent plans), are resource intensive and react slowly to local changes in dynamic environments. Distributed approximate algorithms, where agents plan individually and coordinate with each other locally through consensus protocols, can alleviate many of these issues and have been successfully used to develop real-time conflict-free solutions for heterogeneous networked teams.

An important issue associated with autonomous planning is that many of the algorithms rely on underlying system models and parameters which are often subject to uncertainty. This uncertainty can result from many sources including: inaccurate modeling due to simplifications, assumptions, and/or parameter errors; fundamentally nondeterministic processes (e.g. sensor readings, stochastic dynamics); and dynamic local information changes. As discrepancies between the planner models and the actual system dynamics increase, mission performance typically degrades. The impact of these discrepancies on the overall quality of the plan is usually hard to quantify in advance due to nonlinear effects, coupling between tasks and agents, and interdependencies between system constraints. However, if uncertainty models of planning parameters are available, they can be leveraged to create robust plans that explicitly hedge against the inherent uncertainty given allowable risk thresholds.

This thesis presents real-time robust distributed planning strategies that can be used to plan for multi-agent networked teams operating in stochastic and dynamic environments. One class of distributed combinatorial planning algorithms involves using auction algorithms augmented with consensus protocols to allocate tasks amongst a team of agents while resolving conflicting assignments locally between the agents. A particular algorithm in this class is the Consensus-Based Bundle Algorithm (CBBA), a distributed auction protocol that guarantees conflict-free solutions despite inconsistencies in situational awareness across the team. CBBA runs in polynomial time, demonstrating good scalability with increasing numbers of agents and tasks. This thesis builds upon the CBBA framework to

address many realistic considerations associated with planning for networked teams, including time-critical mission constraints, limited communication between agents, and stochastic operating environments.

A particular focus of this work is a robust extension to CBBA that handles distributed planning in stochastic environments given probabilistic parameter models and different stochastic metrics. The Robust CBBA algorithm proposed in this thesis provides a distributed real-time framework which can leverage different stochastic metrics to hedge against parameter uncertainty. In mission scenarios where low probability of failure is required, a chance-constrained stochastic metric can be used to provide probabilistic guarantees on achievable mission performance given allowable risk thresholds. This thesis proposes a distributed chance-constrained approximation that can be used within the Robust CBBA framework, and derives constraints on individual risk allocations to guarantee equivalence between the centralized chance-constrained optimization and the distributed approximation. Different risk allocation strategies for homogeneous and heterogeneous teams are proposed that approximate the agent and mission score distributions *a priori*, and results are provided showing improved performance in time-critical mission scenarios given allowable risk thresholds.

Thesis Supervisor: Jonathan P. How
Title: Richard C. Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

There are several people I would like to acknowledge as being instrumental to the development of this thesis. First of all I would like to thank my advisor Prof. Jonathan How for his endless support, advice and inspiration. You have helped me grow immeasurably, both as a researcher and as a person, and I am very grateful to have had the privilege of working with you. Your invaluable support and mentorship, not just throughout my research, but in my other endeavors at MIT as well, have made these past four years an amazing experience. I would like to thank my colleague and friend Luke Johnson, who I've had the pleasure of working with on most of my research projects. This thesis would really not have been possible without your continued support, brilliance, and inspiration. To my committee members, Prof. Missy Cummings and Prof. Devavrat Shah, thank you for your support and advice throughout this thesis. To my thesis readers, Prof. Han-Lim Choi and Prof. Julie Shah, thank you for your detailed revisions and insightful feedback. And to all my colleagues and friends at the Aerospace Controls Lab, especially Andrew K., Buddy, Dan, Tuna, Trevor, Kemal, Brandon, Frank, Josh and Vishnu, thanks for your friendship, for your willingness to always help me out with my technical crises, and for making the last four years at ACL such a fabulous experience.

I would also like to thank the Aero/Astro faculty, especially Prof. Karen Willcox, Prof. Dava Newman, Prof. Jeff Hoffman, Prof. Jaime Peraire, Prof. Dave Darmofal and Prof. Ian Waitz, for their endless support of the Women's Graduate Association of Aeronautics and Astronautics (WGA3) and for being so encouraging in general. It has been a truly wonderful experience being part of Aero/Astro and I have learned so much from all of you. To my fellow WGA3 ladies Sunny and Chelsea, thanks for sharing your vision with me and for all your hard work, creativity and inspiration. I have no doubt that you will continue to accomplish many great things. And to the Aero/Astro staff, especially Marie Stuppard, Kathryn Fischer, Beth Marois, Barbara Lechner, Dave Robertson, Dick Perdichizzi, and Bill Litant thanks for always being there to help me out with grad school and other projects at MIT, for always being so willing to do tours and other favours for me, and for all your encouragement, support, and friendship.

I would also like to thank all my dear friends for making these past four years unforgettable, especially the Aero/Astro gang: Sunny, Francois, Arde, Bjoern, Andreas, Leo,

Fabio, Beenie, Amy, Josh, Irene, Namiko, Jeff, and many more, and to my undergrad buddies especially Laura, Cara, and Debbie. Thanks for all your support and friendship, for listening to all my rants, for always being there for me, and for all the wonderful times we've had together. I couldn't have done it without you guys. And most importantly to my family, thank you for always supporting me, loving me, and being there for me. And in particular, to my parents, I'm really grateful for the plentiful opportunities you have given me throughout my life. You have always been a source of inspiration and encouragement and I love you both very much.

This work was supported by MURI (FA9550-08-1-0356). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Algorithms

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

1.1 Motivation

The increased use of robotic agents, such as unmanned aerial vehicles (UAVs) and autonomous ground rovers, has motivated the development of autonomous cooperative task allocation and planning methods. Teams of heterogeneous networked agents are regularly employed in several different types of autonomous missions including intelligence, surveillance, and reconnaissance (ISR) operations [1, 2], environmental disaster relief [146], search and rescue operations [219], fighting forest fires [207], precision agriculture [197], and weather forecasting [84]. Such missions typically involve executing several different activities, sometimes simultaneously, where agents must coordinate and interact with each other to perform the requisite mission tasks. Agents within these networked teams are usually heterogeneous, possessing different resources and capabilities, and some agents are better suited to handle certain types of tasks than others leading to different roles and responsibilities within the mission. For example, UAVs equipped with video can perform search, surveillance, and target tracking tasks, human operators can visually classify targets, monitor system status, and perform supervisory tasks, and ground teams can be deployed to perform rescue operations or engage targets. Figure 1-1 illustrates an example ISR mission scenario involving numerous networked agents performing a variety of search, track, and surveillance tasks, and Figure 1-2 shows a few examples of real systems that are currently used in ISR missions including small and large UAVs and human-in-the-loop operators. Ensuring proper coordination and collaboration between the different agents in the team is crucial to efficient and successful mission execution, motivating the development of au-

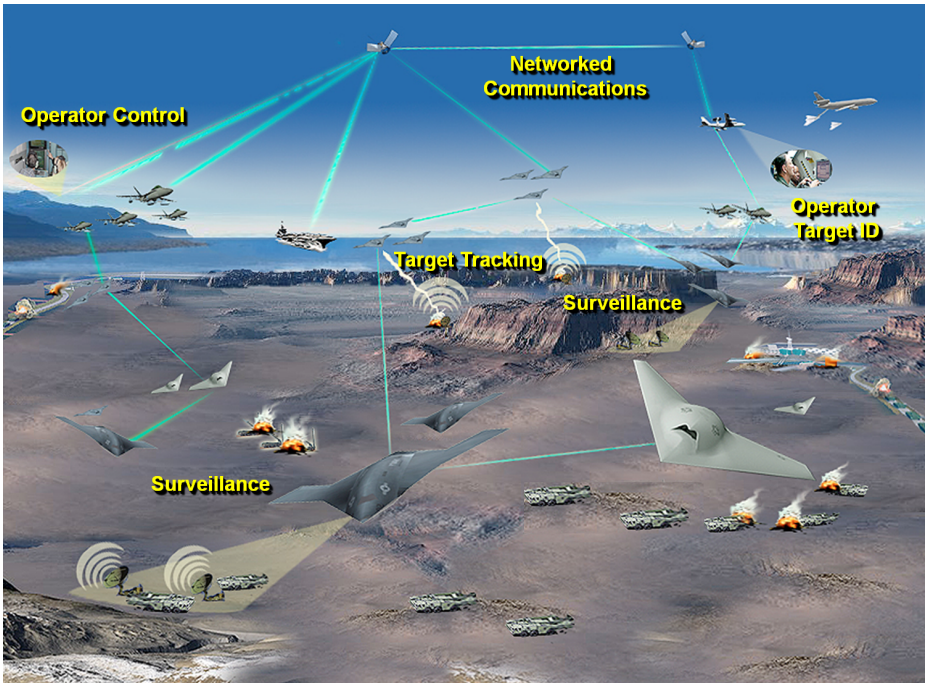


Figure 1-1: Illustration of an example ISR mission scenario involving numerous agents performing a variety of search, track, and surveillance tasks.

tonomous task allocation and planning methods for heterogeneous networked teams. The goal of such planning algorithms is to distribute the required mission tasks amongst the agents so as to optimize overall mission efficiency, and to ensure spatial and temporal synchronization of the team while considering mission costs, available resources and network constraints. Furthermore, the advancement of communication systems, sensors, and embedded technology has significantly increased the value of those solutions that are scalable to larger teams, from dozens to hundreds or even thousands of agents [1, 2]. However, as the number of systems, components, and mission tasks grows, planning for such teams becomes increasingly complex.

(a) Remotely piloted small UAV

(b) Predator UAV

(c) Predator UAV operator

Figure 1-2: Examples of current systems and components comprising ISR missions.

Autonomous mission planning for teams of networked agents has several inherent challenges. Firstly, task allocation involves solving complex combinatorial (mixed-integer) decision problems (NP-hard), which scale poorly and for which optimal solutions are usually computationally intractable for large numbers of agents and tasks [36]. Developing useful agent models that can be embedded within a planning framework is typically a complex endeavor. Agent dynamics are not usually well understood or cannot be well represented, and, as such, the underlying agent models typically involve several simplifying assumptions about the problem structure, agent dynamics, system states and the operating environment. In spite of these simplifications, the resulting models often involve nonlinear and time-varying transition dynamics and constraints that are difficult to handle within a combinatorial optimization framework. This challenging problem is further complicated by realistic mission considerations such as resource limitations (fuel, payload, ordnance, etc), varying communication constraints, cooperative task execution, time-varying score functions, and unknown dynamic environments for which limited *a priori* information is available [71, 114, 174, 220]. Finally, since mission operations usually involve dynamic environments, where agents' situational awareness and underlying models change rapidly as new information is acquired, the planning strategies employed must consist of *computationally efficient* algorithms that can adjust or recompute solutions in real time.

A major limitation associated with deterministically planning for heterogeneous networked teams is that the problem formulation and planning solutions rely on the underlying system models and parameters. In unknown and dynamic environments, these models and parameters are typically uncertain, and deterministic planning methods must therefore use parameter approximations to make decisions (e.g. maximum likelihood estimate of the parameters given the agent's current situational awareness). Discrepancies between these planner models and the actual system dynamics cause degradations in mission performance and solution quality. Furthermore, the impact of these discrepancies on the overall quality of the plan is typically hard to quantify in advance due to nonlinear effects, coupling between tasks and agents, and interdependencies between system constraints. For example, if an agent takes longer than predicted to complete a task, this will not only affect the score obtained for that task, but is also likely to impact the arrival time of his next task. In time-critical missions, this compounding effect will cause deteriorations in the performance of the overall mission. As difficulties arise, agents can replan to reassign the team's as-

sets in light of new information, however, it is likely that the initial imprecise allocations will have caused the team to consume resources (e.g. fuel, power, ordnance) and may have positioned agents such that it is difficult for them to respond effectively to new threats and targets. Thus it is often better to explicitly hedge against the uncertainty, selecting plans that "expect the unexpected" and are less sensitive to modeling errors and imprecise information. In particular, if additional information or models for the planning parameters are available, these can be leveraged to produce plans that exhibit good robustness properties [34, 35]. Examples of parameter uncertainty models could include probabilistic distributions, stochastic metrics such as moments, central moments or bounds, Markov models, Gaussian processes, Bayesian nonparametric models, etc. The tradeoff for this increase in planner performance is a substantial (often exponential) growth in computational complexity, since the planner must now consider the many possible outcomes provided by the uncertainty models in addition to enumerating the planning options.

This thesis presents real-time distributed robust planning strategies that can effectively embed uncertainty models of planning parameters into the score functions, transition dynamics, and constraints, creating plans that hedge against the inherent uncertainty. The next few sections provide further insight into the key technical challenges and current solution methodologies associated with generating such planning strategies.

1.2 Literature Review

There are several research areas that address aspects of the robust planning problem for large heterogeneous networked teams. Some of these are described in the following sections and include cooperative planning strategies for autonomous networked teams, stochastic planning approaches and robust optimization, and efficient inference techniques to represent and sample from complex uncertainty models.

1.2.1 Planning Strategies

Many different methods have been considered for allocating tasks amongst a team of agents. The basic problem can be formulated as a combinatorial optimization mixed-integer program, involving nonlinear dynamics and integer and continuous decision variables. Mixed-integer problems are significantly harder to solve than their linear programming counter-

parts, and exhibit poor scalability since computation time increases exponentially with the problem size [36]. For most problems of interest, optimal solution methods are computationally intractable motivating the development of many approximation techniques [36]. Centralized planning approaches, where a control center plans and distributes tasks to all agents, usually require high bandwidth connections with a ground station (e.g. to transmit received sensor data, and to dispense agent plans), and thus are resource intensive and react slowly to local changes in dynamic environments. Distributed algorithms, where agents plan amongst themselves and coordinate with each other, present several advantages over centralized solutions [51, 63, 148, 202], such as fewer resource requirements and faster reaction times to local information changes. One class of distributed combinatorial planning algorithms involves using auction algorithms augmented with consensus protocols to allocate tasks over a team of agents while resolving conflicting assignments locally among the agents [3, 58, 194]. An example is the Consensus-Based Bundle Algorithm (CBBA) [58], a distributed auction protocol that guarantees conflict-free solutions despite inconsistencies in situational awareness across the team. CBBA runs in polynomial time, demonstrating good scalability with increasing numbers of agents and tasks, and enabling real-time distributed planning for multi-agent networked teams[1].

Although several of these distributed planning algorithms have been successfully demonstrated through simulation and experimentation, their solution quality is dependent on the accuracy of the underlying system models. Since plans generated deterministically are typically rigid and do not allow for much slack in plan execution, performance can degrade substantially if the planning parameters are different than expected, possibly leading to plan infeasibility [144]. Furthermore, since the underlying optimization problem is discrete in nature, it is difficult to predict and quantify in advance the repercussions of plan deviations, motivating the study of robust planning strategies.

1.2.2 Robust and Stochastic Optimization

The problem of robust combinatorial optimization has been extensively explored in the literature, mostly within the context of centralized planning. Stochastic planning techniques have been employed in several applications, ranging from operations research, to robust

[1]This thesis builds upon the CBBA framework to address many realistic considerations associated with planning for networked teams operating in uncertain and dynamic environments, in particular focusing on robustness to communication and parameter uncertainty.

portfolio optimization strategies [74, 143, 226], from dynamic vehicle routing [213], to UAV operations [38, 40, 122, 123, 147, 187, 188]. The airline scheduling community has also explored the robust planning problem in the context of airline operations, focusing on issues such as minimizing the impact of delays on fleet schedules, robust crew and resource allocation, and optimizing runway and taxi operations [15, 17, 18, 59, 125, 128, 129, 136, 200]. While these varied approaches provide valuable insights into the challenges and benefits associated with robust planning, they also highlight several key issues. Firstly, robust planning involves solving large combinatorial optimization problems which scale poorly as the problem size increases, especially when uncertainty must be explicitly accounted for (curse of dimensionality [30]), and most approaches consider centralized solutions which cannot easily be extended to distributed environments. Secondly, developing scoring and constraint functions in the robust problem formulation involves coupling and combining probability distributions, which is usually nontrivial unless limiting assumptions on the underlying probability models are made (e.g. independent identically distributed parameters). Even when analytic expressions for the combined distributions can be derived, stochastic metrics (such as moments or expectations) cannot usually be computed in closed form. Finally, as parameter distributions are updated (as additional information is acquired), the scoring and constraint functions, combined distributions, and planner solution must be completely recomputed to account for the new information.

Many of the stochastic planning algorithms employed in the above examples involve maximizing the expected-value or average system performance [28]. In missions where stronger performance guarantees than optimizing average scores are required, a stochastic metric to mitigate the worst-case possible system performance can be used instead. Classical "robust" formulations, that optimize worst-case system performance, have been considered for integer programming problems [25, 34, 43, 154–156]. However, these approaches typically involve making several limiting assumptions about the problem formulation and the form of the uncertainty, in order to maintain analytic and computational tractability (i.e. linear program formulation, bounded and symmetric or ellipsoidal uncertainty models, independence between parameter distributions, etc.), which restrict the scope of the solution methodologies making generalization to more complex realistic mission scenarios difficult. Additionally, mitigating worst-case system performance is often too conservative, motivating the development of probabilistic planning approaches that reduce the risk of failure to

within a predefined threshold [120, 121, 190]. In [35], the degree of conservatism can be controlled by setting the maximum number of parameters that take their worst-case values, and the problem can be converted into an equivalent deterministic problem and solved using standard approaches. Their solution provides robustness bounds for each constraint being violated, but does not address the robustness of the overall solution. An alternative chance-constrained formulation, presented in [55], guarantees system performance within an allowable risk threshold. However, the coupling of the uncertainty under this formulation limits the applicability of the approach, especially when uncertainty is present in the constraints, and the computational requirements associated with solving this optimization are much higher than the previous approaches. Recent work in [144] extends some of these classical formulations to address robustness of the total solution and computational tractability of the algorithms, however, the approach in [144] still assumes linear cost and constraint formulations and specific forms of parameter uncertainty, and cannot be easily extended to support the types of system dynamics and uncertainty models of interest in this thesis (time-varying, nonlinear, nonmonotonic score functions, and nonsymmetric or unbounded distributions, etc).

1.2.3 Representing Uncertainty

As mentioned in the previous section, working with uncertainty models within an optimization framework requires combining distributions and evaluating stochastic metrics, for which analytic expressions cannot usually be derived unless the underlying distributions are of very specific types (e.g. independent identically distributed random variables, Gaussian, exponential or Bernoulli distributions, etc). This limits the practical applicability of using a typical Bayesian framework that, while theoretically sound, cannot be implemented directly except in a few very special cases, since the integrals required for most of the essential Bayesian calculations (normalization, marginalization, computing moments, etc) are often intractable [11, 67, 86]. To address these inference related issues, sampling algorithms, that approximate complex distributions with a finite number of representative samples, have been extensively explored. These algorithms, commonly known as Monte Carlo methods, were originally developed in the 40's [150] and have been widely used to efficiently solve complex high-dimensional problems in a variety of fields such as statistics, econometrics, decision analysis, physics and machine learning [11, 86]. Examples of com-

monly used Monte Carlo methods include rejection sampling, importance sampling and Markov Chain Monte Carlo (MCMC). MCMC methods in particular, have been widely used in the literature, due to their simplicity, computational efficiency, and ability to represent very complex high-dimensional systems [11]. The most popular MCMC method is the Metropolis-Hastings algorithm [97] and consists of the following steps: initialize the system, draw a sample from some proposal distribution which is a function of the current state, use the sample as the new state with some acceptance probability. A key feature of the algorithm is that the acceptance probability involves a ratio of the proposal and actual probability distributions and therefore does not require computing the normalization constant of either distribution, a property that can be exploited by many algorithms involving Bayesian propagation. The Metropolis-Hastings algorithm is the most general form of MCMC and it can be shown that several of the popular MCMC algorithms (Gibbs sampling, simulated annealing, independence sampling, symmetric random walk Metropolis, etc.) are special cases of the Metropolis-Hastings algorithm that use specific forms of proposal distributions and acceptance probabilities [11, 67].

The main issues associated with MCMC algorithms are that they usually require a large number of samples to obtain accurate representations, and algorithm convergence and performance are highly dependent on the choice of proposal distribution. It is difficult to design a proposal distribution that is easy to sample from yet provides a realistic representation of the underlying complex system, and this approach is typically empirical and problem dependent. Furthermore, it is difficult to assess convergence to stationarity and errors in the estimates, since the very nature of the problem does not typically allow for the calculation of a baseline truth [73]. But in spite of these difficulties, MCMC methods are promising solutions to problems where there are often no other alternatives, and if specific distribution forms are available (e.g. conditional conjugate distributions, etc), then algorithm convergence is typically fast and accurate. Another advantage of sampling algorithms, is that individual samples can often be updated online using Bayesian updates or transition dynamics as the environment changes or more information is acquired (e.g. sequential Monte Carlo or particle filtering algorithms [11, 189]), allowing for a reusable framework that does not require resampling all the particles every time the uncertainty models are updated. Stochastic planning algorithms that use sampling methods for inference have been developed mainly for continuous optimization problems (e.g. trajectory optimization, collision

avoidance, etc), however, extensions to discrete optimization problems such as task allocation are nontrivial.

A key design decision, that drastically impacts the performance of the stochastic planning and sampling algorithms employed, consists of choosing the underlying uncertainty models to represent the stochastic parameters of interest. This typically involves a tradeoff between modeling the uncertain distributions as accurately as possible, and using uncertainty representations that work well within the planning framework. For example, if the stochastic planning algorithms involve computing metrics over sums of random variables, then only very few distributions under limiting assumptions allow these computations to be performed analytically in closed form (e.g. independent identically distributed random variables with Gaussian, exponential or Bernoulli distributions), or if distribution parameters must be updated in real time given new data acquired, then using conjugate distributions allows for convenient recursive updates within the planner (e.g. gamma, Dirichlet, Gaussian), but generic distributions typically require recomputing the parameters given the new data. The use of sampling algorithms within a stochastic planning framework allows for a wider variety of uncertainty representations, alleviating several of these limitations, however the performance and computational efficiency of these algorithms is highly dependent on the uncertainty representations used (choice of proposal distribution, parameter dimensionality, number of modes, etc). A special example is the Gibbs sampler, where the use of conditional conjugate distributions (such as Dirichlet) drastically improves algorithm performance, allowing for very efficient measurement updates [67]. The main challenges and questions to consider when developing the underlying uncertainty representations are: how to balance the tradeoff between choosing an efficient model versus realistically representing the system dynamics, how accurately does the model represent the system given the current data, how efficiently can the model be updated, and how well can the model adapt as new information is acquired.

1.2.4 Remaining Challenges

The goal of this research is to develop *real-time distributed robust planning strategies* that can effectively embed uncertainty models of planning parameters into the score functions, transition dynamics, and constraints, creating plans that hedge against the inherent uncertainty. Although the current literature described in Section 1.2.2 provides many insights and

useful lessons, the inherent assumptions adopted in most of the approaches (e.g. linearity, independent homogeneous distributions, etc.) limits their applicability when planning for more general realistic mission scenarios. The dynamic models associated with networked teams of heterogeneous agents executing missions with temporal and spatial constraints typically involve nonlinear time-varying functions, and the uncertainty models of interest consist of nonsymmetric dependent distributions for which analytic expressions for sums of random variables are usually not available. Due to these limitations, this thesis explores the development of computationally efficient planning strategies that are flexible enough to allow for heterogeneous agent dynamics and varied uncertainty representations.

There are several key challenges associated with this problem. The first involves developing methods to embed general models of parameter uncertainty into the planning framework. This work explores numerical methods to efficiently represent complex uncertainty models, combined distributions, and stochastic metrics of interest through sampling approaches (e.g. expectations, percentile thresholds, etc). Furthermore, to enable real-time computationally efficient planning, this work employs polynomial-time approximation algorithms that incrementally build solutions (e.g. sequential greedy) rather than optimal algorithms that enumerate all possible assignments (NP-hard). The main challenge therein is to ensure that the uncertainty of the total solution is represented appropriately when making these incremental decisions. Furthermore, when planning for large-scale networked teams operating in dynamic environments it is advantageous to consider distributed planning strategies, however, most of the distributed approaches described in Section 1.2.1 cannot be trivially extended to account for parameter uncertainty and robustness. This work presents a robust distributed planning framework that extends state-of-the-art distributed algorithms to include uncertainty models. Since most of these distributed algorithms involve consensus on plans between the different agents, the key challenge here lies in ensuring that the uncertainty can be represented locally within each agent's planning process, while still guaranteeing convergence of the distributed algorithm. This thesis extends the Consensus-Based Bundle Algorithm (CBBA) presented in [58] and leverages its converge guarantees under varying agent situational awareness to ensure that agents can create robust plans locally given their own uncertainty representations. The next section provides details on the specific extensions of CBBA and contributions of this thesis.

1.3 Thesis Contributions

This thesis addresses the problem of real-time robust distributed planning for multi-agent networked teams operating in uncertain and dynamic environments. In particular, several extensions and variants to the baseline CBBA algorithm presented in [58] are proposed and discussed, enabling distributed real-time planning in time-critical, communication-limited, and uncertain environments. The specific contributions of this thesis are described as follows:

1. This thesis extends CBBA to handle time-critical mission considerations, where time-varying score functions can be optimized within the CBBA algorithmic framework to enable dynamic planning for agents and tasks with specific timing constraints (e.g. task time-windows of validity, time-varying rewards for time-critical tasks, agent velocities). In particular, the CBBA with Time-Varying Score Functions algorithm proposed in Section 4.2 modifies the bundle construction process of CBBA to explicitly optimize task execution times as well as agent assignments, enabling both spatial and temporal coordination of multi-agent teams in dynamic mission scenarios. The algorithm performance was validated through simulations, and a real-time replanning architecture was designed and implemented to enable real-time experiments for heterogeneous networked teams. Flight test experiments involving multi-agent dynamic search and track missions were performed in an indoor flight test facility at the MIT Aerospace Controls Lab using heterogeneous teams of quadrotor UAVs and robotic ground vehicles, demonstrating the real-time applicability of the distributed planning algorithms.

2. This thesis extends the CBBA planning framework to enable conflict-free distributed planning in the presence of network disconnects due to communication-limited operating environments. The proposed approach, described in Section 4.3, employs a local distributed task space partitioning strategy, where the sets of tasks available to the different agent sub-networks are disjoint, thus ensuring conflict-free solutions. Simulation and experimental flight tests validated the proposed algorithms, showing improved performance over the baseline CBBA algorithm, but with lower communication and computational overhead than centralized strategies which require *a priori* task space partitioning at every replan iteration.

3. In Section 4.4, we develop a distributed cooperative planning algorithm that builds upon the CBBA framework to enable agents to maintain network connectivity in communication-limited environments. The algorithm, named CBBA with Relays, guarantees network connectivity for agents performing tasks that require a live connection (e.g. video streaming), by locally incentivizing under-utilized agents to act as communication relays for other agents within a distributed framework. The proposed algorithm explicitly handles the joint network connectivity constraints through a local distributed network prediction phase, and cooperation between agents is enabled through the local creation of relay tasks by the agents that require a network connection. The CBBA with Relays algorithm is guaranteed to converge, runs in real-time, and guarantees network connectivity while tasks are being executed. The algorithm was validated through simulation, and indoor and outdoor flight test experiments, demonstrating real-time applicability.

4. In Chapter 5, we extend the CBBA with Time-Varying Score Functions algorithm of Section 4.2 to explicitly account for robustness in the planning process. A real-time distributed robust planning framework, named Robust CBBA, is proposed, which can leverage probabilistic models of planning parameters and different distributable stochastic metrics to hedge against parameter uncertainty. The algorithm employs sampling approaches to compute agent path scores given different stochastic metrics within the CBBA bundle construction process, enabling polynomial-time algorithm convergence. The Robust CBBA framework leverages a recent submodular extension of CBBA proposed by Johnson [106] to guarantee distributed algorithm convergence given different stochastic metrics, and uses the convergence guarantees of CBBA under varying situational awareness to allow agents to individually construct their robust plans given local uncertainty representations. The Robust CBBA algorithm was implemented using two stochastic metrics, the expected-value metric and the worst-case stochastic metric, and used to plan for heterogeneous multi-agent teams performing search and track missions in uncertain environments. Simulation results are provided demonstrating real-time applicability, and showing that Robust CBBA improves performance over the baseline CBBA algorithm and achieves results similar to centralized planning strategies, validating the distributed approach.

5. In Chapter 6, we extend the Robust CBBA framework proposed in Chapter 5 to optimize performance in environments where low probability of failure is required. The approach uses a chance-constrained stochastic metric that provides probabilistic guarantees on achievable mission performance given allowable risk thresholds. A distributed approximation to the chance-constrained metric is proposed to enable the use of Robust CBBA in these risk-aware environments, and constraints on individual risk allocations are derived to guarantee equivalence between the centralized chance-constrained optimization and the distributed approximation. Different risk allocation strategies for homogeneous and heterogeneous teams are proposed that approximate the agent and mission score distributions *a priori*, and results are provided comparing the performance of these in time-critical mission scenarios. The distributed chance-constrained CBBA algorithm was validated through simulation trials, and the results showed improved performance over baseline CBBA and over worst-case conservative planning strategies given allowable risk thresholds. Furthermore, the distributed chance-constrained approximation algorithm proposed in Chapter 6 achieved similar results to those obtained by centralized chance-constrained methods, validating the distributed approximation.

The next section describes the layout of the thesis.

1.4 Thesis Layout

The remainder of this thesis is organized as follows. Chapter 2 defines the task allocation problem statement, the time-varying version of the task allocation problem statement, and the specific task allocation formulation considered in this thesis. It also provides a discussion on different robust metrics that can be used in stochastic planning environments and formalizes the language and variables used throughout this thesis. The chapter ends by discussing common solution approaches to multi-agent planning problems explored in the literature within a centralized framework.

Chapter 3 provides background on distributed planning, describing various architectural decisions that must be addressed when implementing real-time planning algorithms for autonomous multi-agent teams. In particular, insights are provided on when centralized, distributed, and decentralized architectures can be used given the communication

infrastructure and available computational resources of the multi-agent system. Different considerations and challenges associated with distributed planning are discussed, and algorithms that can be utilized within distributed planning frameworks are identified. One particular algorithm, the Consensus-Based Bundle Algorithm (CBBA) [58], is highlighted as an algorithm which is well-suited to dynamic real-time planning environments, and provides the foundation for the remainder of the work in this thesis.

Chapter 4 describes the Consensus-Based Bundle Algorithm (CBBA) developed by Choi et al. [58], and presents key extensions and variants developed in this thesis to address dynamic mission planning in realistic environments. Section 4.1 describes the baseline CBBA algorithm proposed in [58]. Section 4.2 proposes an extension to CBBA that enables optimization given time-varying score functions and dynamic mission scenarios (CBBA with Time-Varying Score Functions). Section 4.3 presents strategies to ensure conflict-free solutions in the presence of network disconnects through local task space partitioning methods. And, Section 4.4 proposes a cooperative planning algorithm (CBBA with Relays), that builds upon the baseline CBBA framework to enable cooperative mission execution in communication-limited environments through the use of relay tasks.

Chapter 5 describes how uncertainty models of planning parameters can be leveraged within the distributed planning framework to create robust plans that explicitly hedge against the uncertainty. A stochastic planning extension to the distributed CBBA algorithm is proposed (Robust CBBA), describing how expected-value and worst-case stochastic metrics can be embedded within the planning framework using numerical sampling techniques. Finally results for homogeneous and heterogeneous networked teams are provided demonstrating performance improvements for multi-agent teams operating in uncertain and dynamic environments.

Chapter 6 explores the use of the chance-constrained stochastic metric, which provides more flexibility over the conservatism of the solution, and discusses the different distributed planning considerations associated with risk-aware planning. A distributed approximation of the centralized chance-constrained optimization is proposed which can be used within the Robust CBBA framework, and different methods to allocate risk amongst the different agents are discussed. Finally, simulation results for homogeneous and heterogeneous networked teams are provided, validating the proposed approach. Chapter 7 presents a summary of the thesis contributions and directions for future work.

Appendix A provides detailed derivations for the different risk allocation strategies proposed in Chapter 6, discussing how risk can be allocated amongst the different agents given homogeneous and heterogeneous teams. And finally, Appendix B describes joint work with Cornell University, where a distributed information-based planning framework was created to address the issue of how to obtain better data to improve models and reduce uncertainty. The proposed approach involves a novel planning and estimation architecture, where the goal of maximizing information is a primary objective for each of the algorithms at every step, producing a cohesive framework that strategically addresses the main mission objectives.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Problem Statement

This chapter defines the generic problem formulation and a few key variants, formalizing the language and variables used throughout this thesis. It also discusses common solution approaches to multi-agent planning problems.

2.1 Problem Formulations

2.1.1 Multi-Agent Task Allocation

Given a team of N_a agents and set of N_t tasks, the goal of the task allocation algorithm is to find a matching of tasks to agents that maximizes some global reward (see Figure 2-1). The global objective function for the mission is given by a sum over local objective functions for each agent-task pair, which are in turn functions of the agent assignments and the set of planning parameters. This task assignment problem can be written as the following integer program,

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{i=1}^{N_a} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\mathbf{x}, \boldsymbol{\theta}) \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{x}, \boldsymbol{\theta}) \leq \mathbf{b} \\ & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{2.1}$$

where the decision variable $\mathbf{x}$ is the global assignment comprised of all agent-task pairings x_{ij} denoting whether agent i is assigned to task j (i.e. $\mathcal{X} \triangleq \{0,1\}^{N_a \times N_t}$). In the above formulation, $\boldsymbol{\theta}$ is a set of planning parameters that affect the score calculation (e.g. fuel

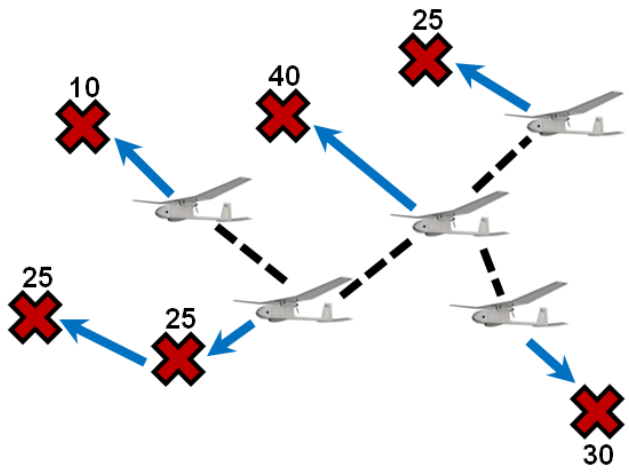


Figure 2-1: Task allocation example for a networked team of UAVs performing target search and track tasks at different locations. Blue arrows denote the path of each UAV, and the red X's represent task locations with associated score values.

costs, task rewards, agent and task positions, etc.), and $\mathbf{c}_{ij}$ is the reward agent i receives for task j given the overall assignment $\mathbf{x}$ and parameters $\boldsymbol{\theta}$. The optimization problem is subject to a set of constraints, $\mathbf{G}(\mathbf{x}, \boldsymbol{\theta}) \leq \mathbf{b}$, where $\mathbf{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_{N_c}]^T$ and $\mathbf{b} = [b_1, \ldots, b_{N_c}]^T$ defining N_c possibly nonlinear constraints of the form $\mathbf{g}_k(\mathbf{x}, \boldsymbol{\theta}) \leq b_k$, capturing vehicle transition dynamics, resource limitations, etc. This generalized problem formulation can accommodate several different design objectives and constraints commonly used in multi-agent decision making problems. Some examples include search and surveillance missions where $\mathbf{c}_{ij}$ could represent the value of acquired information and the constraints $\mathbf{g}_k$ could capture fuel limitations and no-fly zones, or human-robot missions where $\mathbf{c}_{ij}$ and $\mathbf{g}_k$ could capture interactions between operators and autonomous agents.

An important observation is that, in Eq. (2.1), the scoring and constraint functions are explicitly dependent upon the decision variables in $\mathbf{x}$, making this complex combinatorial decision problem very difficult to solve in general (NP-hard) due to the inherent system interdependencies [36]. Examples of this type of multi-agent multi-task allocation problem include the well-studied Traveling Salesman Problem (TSP) and the Dynamic Vehicle Routing Problem (DVRP) [213], which are widely recognized as complex, computationally-intensive optimization problems.

2.1.2 Multi-Agent Task Allocation With Time-Varying Score Functions

It is often of interest to consider dynamic task allocation problems where rewards can change over time. For example, some tasks, such as scheduled landings at airports, can have specific time-windows of validity, or other tasks, such as rescue operations, may be time-critical favouring earlier task service times (see Figure 2-2 for examples of time-varying score functions). In these types of time-dependent scenarios, the task allocation framework can be extended to include selection of task service times as well as task allocations. This can be formulated as the following mixed-integer program,

$$\begin{aligned} \max_{\mathbf{x},\boldsymbol{\tau}} \quad & \sum_{i=1}^{N_a}\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\mathbf{x},\boldsymbol{\tau},\boldsymbol{\theta}) \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{x},\boldsymbol{\tau},\boldsymbol{\theta}) \leq \mathbf{b} \\ & \mathbf{x} \in \mathcal{X} \\ & \boldsymbol{\tau} \in \mathcal{T} \end{aligned} \tag{2.2}$$

where $\boldsymbol{\tau}$ is an additional set of real-positive decision variables τ_{ij} indicating when agent i will execute its assigned task j, and $\tau_{ij} = \emptyset$ if task j is not assigned to agent i (i.e. $\mathcal{T} \triangleq \{\mathbb{R}^+ \cup \emptyset\}^{N_a \times N_t}$). In this formulation the constraints are functions of the task execution times as well, $\mathbf{g}_k(\mathbf{x},\boldsymbol{\tau},\boldsymbol{\theta}) \leq b_k$, allowing time-varying constraints to be represented in the optimization (e.g. time-varying agent dynamics, temporal constraints and dependencies between tasks, etc.). Examples of time-varying multi-agent multi-task allocation problems in the literature include DVRP with time-windows (DVRPTW) [213], servicing impatient customers with strict service time-windows [167], and developing MILP frameworks and hybrid models for DVRPTW problems [71, 110].

The above formulations specified in Eqs. (2.1) and (2.2) are generic enough to accommodate explicit agent cooperation through coupled score functions. As an example, if a task requires two (or more) agents for successful completion (e.g. remotely piloted UAV's require assigning the UAV asset and the human operator simultaneously), then the two agents i and k must collaborate to perform task j, thus x_{ij} and x_{kj} are both 1, and $\mathbf{c}_{ij}$ and $\mathbf{c}_{kj}$ depend upon both x_{ij} and x_{kj} being assigned. Similarly, tasks may have dependencies between them. For example, a rescue mission might require a UAV search task to locate the victim and a subsequent ground convoy task to perform the rescue. In this framework,

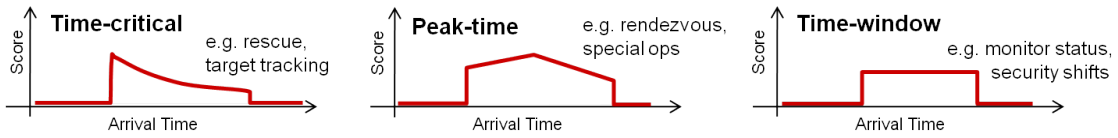


Figure 2-2: Examples of time-varying score functions for different types of tasks.

the ground convoy rescue task m would only receive a positive score if the UAV search task j is assigned as well, thus for any agent i, $\mathbf{c}_{im}$ is dependent on both x_{im} and $x_{\star j}$ being assigned[1]. These types of coupled problems are very complex and difficult to solve, especially in distributed environments, due to all the inherent interdependencies in the system. Furthermore, because of this extreme generality, the cardinality of the state-space in Eq. (2.2) is uncountably infinite and thus is a very difficult space to search, even approximately. The following section describes some simplifying assumptions regarding task independence and agent independence that are typically added to the above formulations to make solution approaches tractable. For further discussion on cooperative planning through coupled constraints see Whitten et al. [220].

2.1.3 Simplifying Assumptions and Constraint Specifications

The first assumption employed in this thesis is that every task may be assigned to at most one agent (i.e. two or more agents cannot be assigned to the same task). This is a standard assumption often made in the task allocation literature [58], and is beneficial for the optimization framework because it has the effect of dramatically reducing the cardinality of $\mathcal{X}$. This task service uniqueness constraint is formalized as

$$\begin{aligned} &\sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\ &x_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \end{aligned} \tag{2.3}$$

where $\mathcal{I} \triangleq \{1,\ldots,N_a\}$ and $\mathcal{J} \triangleq \{1,\ldots,N_t\}$ are the index sets that iterate over agents and tasks respectively. The constraint in Eq. (2.3) ensures that the algorithm returns a *conflict-free* matching of tasks to agents that maximizes the global reward[2]. Even though this constraint explicitly prohibits multiple agents from performing the same task, it still

[1] In this notation $\star$ is used to represent any agent. In other words, the score that agent i receives for selecting task m is dependent on whether task j is assigned (possibly to another agent) or not.

[2] An assignment is said to be *conflict-free* if each task is assigned to no more than one agent.

allows for cooperation given some creative task construction. For example, some approaches like posting multiple tasks at the same location and time (implying cooperation will take place) are a brute force way to establish cooperation at a task location. Other approaches encode this cooperation explicitly as more complicated coupling constraints (see Whitten et al. [220]).

The second assumption employed is that the scores agents receive for their tasks are independent of other agents' assignments, locations, and plans[3]. Therefore, agent i's score is a function of $\mathbf{x}_i \triangleq \{x_{i1}, \ldots, x_{iN_t}\}$, a subset of the global assignment $\mathbf{x}$ denoting the task assignments for agent i, and $\boldsymbol{\tau}_i \triangleq \{\tau_{i1}, \ldots, \tau_{iN_t}\}$, a vector of corresponding task execution times for agent i which is a subset of $\boldsymbol{\tau}$. Using this assumption, the global objective function for the mission can be written as a sum over local objective functions for each agent, where each local reward is determined as a function of the tasks assigned to that agent $\mathbf{x}_i$, the times at which those tasks will be executed $\boldsymbol{\tau}_i$, and the set of planning parameters $\boldsymbol{\theta}$. The expression for the global objective function is given by

$$\max_{\mathbf{x},\boldsymbol{\tau}} \quad \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\mathbf{x}_i, \boldsymbol{\tau}_i, \boldsymbol{\theta}) \right)$$

A third simplifying assumption that would be useful to reduce the coupling between decision variables involves task independence, where the score achieved for each task is independent of the completion of all other tasks and locations of all other agents. Unfortunately, in time-varying domains, tasks are not usually independent, since what an agent does prior to arriving at a task potentially impacts the agent's arrival time at that particular task. It is useful in the optimization, however, to encode this temporal task dependence into the task execution decision variables explicitly (i.e. τ_{ij} is a function of the tasks agent i does prior to executing task j). This enables the reward function to be decoupled so that the only two things that affect the score of the task are the capabilities of the agent servicing the task (including the availability of the agent due to commitments of servicing other tasks) and what time the task is actually serviced. The local agent-task scores $\mathbf{c}_{ij}$ thus become functions of τ_{ij} and $\boldsymbol{\theta}$ only. Using these simplifications the global objective

[3]This assumption is employed throughout most of the algorithms used in this thesis, however, Section 4.4 considers a more complex problem that involves coupling between agents' assignments. In particular, in Section 4.4, task scores are dependent on network connectivity, which is a function of the agents' positions over time. The CBBA with Relays algorithm proposed in that section further illustrates how complex it is to explicitly include cooperation and coupling between agents in distributed planning environments.

function can be written as

$$\max_{\mathbf{x},\boldsymbol{\tau}} \quad \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij} \right) \tag{2.4}$$

where the new variable $\mathbf{p}_i$ is the *path* agent i takes given assignment vector $\mathbf{x}_i$. The path $\mathbf{p}_i \triangleq \{p_{i1}, \ldots, p_{i|\mathbf{p}_i|}\}$ is an ordered sequence of tasks composed of elements $p_{in} \in \mathcal{J}$ for $n = \{1, \ldots, |\mathbf{p}_i|\}$, where the k^{th} element, $p_{ik} \in \mathcal{J}$, is the index of the task that agent i conducts at the k^{th} point along the path.

Note that implicit in the task allocation optimization is a path-generation problem which further exacerbates the computational complexity of the combinatorial optimization (i.e. a full solution to the task allocation optimization not only specifies which tasks agents will perform, but also in what order they will execute these). Typically path generation involves finding the most efficient order in which to execute the tasks specified by $\mathbf{x}_i$. For time-varying domains this involves solving

$$\max_{\boldsymbol{\tau}_i,\mathbf{p}_i} \quad \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij}$$

which can be thought of as a two-step process,

$$\max_{\mathbf{p}_i} \left(\max_{\boldsymbol{\tau}_i} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij} \right)$$

where the inner optimization is solved to obtain the optimal task service times $\boldsymbol{\tau}_i^\star$ for each given path $\mathbf{p}_i$, and the outer optimization iterates over all possible paths $\mathbf{p}_i$ to obtain the best path $\mathbf{p}_i^\star$. In most planning algorithms, conservative estimates of the maneuvering capabilities of the agents are typically used to make predictions about routes and timing [22, 179, 195]. Most algorithms in the literature employ path approximations, such as straight-line paths or Dubin's car paths, and the accuracy of the timing and cost estimates used in the planner depend upon these approximation models, possibly leading to poor performance if the actual optimized paths vary greatly from those assumed in the task allocation process.

Finally, it is sometimes useful to impose a maximum path length for each agent, L_i, limiting the amount of tasks that can be assigned to that agent. The maximum path length

constraint can be written as

$$\sum_{j=1}^{N_t} x_{ij} \leq L_i, \quad \forall i \in \mathcal{I} \tag{2.5}$$

and can be used to simulate resource constraints (e.g. fuel, ordnance) instead of explicitly accounting for these in the optimization, thus further simplifying the problem statement. More importantly, this constraint can be imposed as an artificial constraint to help with computation and communication efficiency in the planning algorithms, and is often paired with an implicit receding time horizon to guarantee that an agent does not commit to tasks too far in the future, exhausting its path length.

Similar simplifications regarding agent and task independence can be introduced in the constraint formulations, $\mathbf{G}(\mathbf{x}, \boldsymbol{\tau}, \boldsymbol{\theta}) \leq \mathbf{b}$, to reduce the complexity and coupling associated with the multi-agent optimization. This work assumes that there are no constraints other than those imposed by conflict-free assignments and maximum path lengths as described above. Combining the assumptions and simplifications specified in Eqs. (2.3), (2.4) and (2.5) gives the following problem formulation for multi-agent task allocation in time-varying environments which will be used throughout this thesis:

$$\begin{aligned}
\max_{\mathbf{x},\boldsymbol{\tau}} \quad & \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij} \right) \\
\text{s.t.} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_i, \quad \forall i \in \mathcal{I} \\
& \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\
& x_{ij} \in \{0, 1\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \\
& \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J}
\end{aligned} \tag{2.6}$$

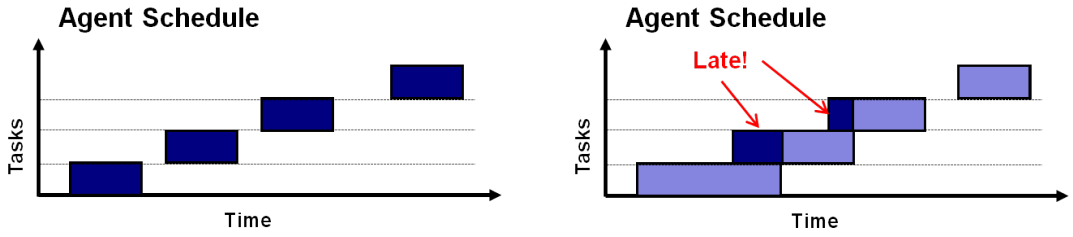
An advantage of formulating the problem as in Eq. (2.6) is that distributing the computation across multiple agents becomes significantly easier. Given a global assignment $\mathbf{x}$, composed of individual agent assignments $\mathbf{x}_i$, each agent can optimize its own path $\mathbf{p}_i$ and select its own task service times $\boldsymbol{\tau}_i$. Thus the primary source of distributed coupling in the task allocation problem is restricted to the choice of the assignment vector $\mathbf{x}$ and the conflict-free constraint.

Even with the above simplifications, this optimization problem remains a complex combinatorial decision problem, with many nonlinearities and interdependencies, for which optimal solution approaches are NP-hard [36]. Furthermore, the planning process depends on the system parameters $\boldsymbol{\theta}$, which are usually assumed to be deterministic. However, in realistic missions, the true values of the planning parameters are typically unknown and only approximations are available (estimates, distributions, etc.). Given stochastic planning parameters, the planning process must account for the uncertainty in $\boldsymbol{\theta}$ in Eq. (2.6), further exacerbating computational intractability [31]. The next section provides details on how to model this uncertainty within the planning framework.

2.1.4 Planning in Uncertain Domains

An important issue associated with planning for networked multi-agent teams is that planning algorithms rely on underlying system models, which are often subject to uncertainty. This uncertainty can result from many sources including: inaccurate modeling due to simplifications, assumptions, and/or parameter errors; fundamentally nondeterministic processes (e.g. sensor readings, stochastic dynamics); and dynamic local information changes. Differences between planner models and actual system dynamics cause degradations in mission performance. Furthermore, the impact of these discrepancies on the overall quality of the plan is typically hard to quantify in advance due to nonlinear effects, coupling between tasks and agents, and interdependencies between system constraints. For example, longer-than-expected task service times not only affect the scores received for those tasks, but also impact the arrival times of subsequent tasks in an agent's path (see Figure 2-3). These types of propagation effects can be catastrophic in certain environments (e.g. time-critical missions), however, if uncertainty models of planning parameters are available they can be leveraged to create robust plans that explicitly hedge against the inherent uncertainty to improve mission performance.

For example, consider the problem definition introduced in Eq. (2.6) but where the planning parameters $\boldsymbol{\theta}$ are random variables. Assume that a model of the uncertainty is available, where $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and is distributed according to the joint probability density function (PDF), $\mathbf{f}(\boldsymbol{\theta})$. Stochastic planning algorithms can use the information provided in $\mathbf{f}(\boldsymbol{\theta})$ to create plans that explicitly account for the *variability* and *coupling* of the uncertain parameters in the score functions $\mathbf{c}_{ij}$. There are several metrics that can be used to account



(a) Proposed Schedule (b) Actual Schedule

Figure 2-3: Illustration of task coupling given uncertain task durations. Figure (a) shows the agent's proposed schedule. Figure (b) shows the agent's actual achieved task durations, where the longer-than-expected duration of the first task impacted the arrival times for the second and third tasks as well. In time-critical missions, this would result in lower-than-expected task scores for all three impacted tasks.

for uncertainty in the planning formulation. Perhaps the most common approach is to maximize the expected mission performance [28], where the objective function from Eq. (2.6) becomes,

$$\max_{\mathbf{x},\boldsymbol{\tau}} \quad \mathbb{E}_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta}) \; x_{ij} \right) \right\} \tag{2.7}$$

Note that optimizing Eq. (2.7) is *not* the same as deterministically planning using the mean values of uncertain planning parameters $\bar{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{\theta}}\{\boldsymbol{\theta}\}$, which can often lead to poor planning performance since the problem formulation fails to capture the nontrivial coupling of uncertainty in scores, dynamics and constraints. This is especially problematic when scores are coupled, and can lead to biased predictions that drastically misrepresent the actual expected performance.

While optimizing Eq. (2.7) provides a plan that maximizes the expected performance of the system, an actual single run execution of this best expected plan is still subject to the uncertainty in the environment, and may result in a relatively poor plan (worse than expected) with some non-zero probability. If the current mission tolerance to failure is very low, a more conservative planning objective involves maximizing the worst-case scenario (sometimes referred to as *robust* in the literature [33]),

$$\max_{\mathbf{x},\boldsymbol{\tau}} \quad \min_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta}) \; x_{ij} \right) \right\} \tag{2.8}$$

Optimizing Eq. (2.8) guarantees that the plan execution will result in a score *no worse* than that predicted by the algorithm, however, in general, execution of the robust plan typically leads to scores higher than the worst case, and for missions of general interest, maximizing this worst-case lower bound is usually far too conservative.

Several robust optimization and stochastic optimization methods, that find the best solution within some predefined risk threshold, have been developed to mitigate this issue of conservatism [33, 157]. One such approach involves optimizing a risk-adjusted expected performance, where a risk function $R(\mathbf{c}_{ij})$ biases the original cost function $\mathbf{c}_{ij}$ towards more conservative solutions to account for the acceptable level of risk. Another approach is to bound the domain of the uncertainty set $\boldsymbol{\theta}$ to be within certain ranges, $\boldsymbol{\theta} \in [\boldsymbol{\Theta}_{\min}, \boldsymbol{\Theta}_{\max}] \subset \boldsymbol{\Theta}$ (e.g. ellipsoids for specified confidence intervals [24]), or to take on a set of discrete representative values $\boldsymbol{\theta} \in [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k] \subset \boldsymbol{\Theta}$ (e.g. by constructing representative uncertainty sets [32, 56]), thus limiting the support of the uncertain parameters. Classical robust convex optimization techniques can then be used to solve the resulting approximate problem [25, 31]. Although these approaches provide methods to control the conservatism or risk associated with the planning solution, there are a few issues which makes their practical implementation difficult. Firstly, it is not usually obvious how to design the risk functions $R(\mathbf{c}_{ij})$ or the bounded uncertainty sets for $\boldsymbol{\theta}$, and the selection of these is typically problem specific, time consuming and ad-hoc. A more serious issue, however, is that when a global risk threshold is available, the metric of interest is the *cumulative* risk of the total solution, not the individual parameter or task risks, and it is difficult to quantify how these individual parameter bounds will affect the risk of the global solution. Nonlinearities in the cost functions, complex variable coupling and interdependencies, and discrete optimization effects, often affect the solution in unpredictable ways, and it is therefore hard to ensure that the total mission outcome is within the desired risk threshold by simply bounding the support of the random variables individually.

An alternative approach that guarantees that the global mission performance will be within a certain risk threshold is the chance-constrained formulation [45, 66, 157],

$$\begin{aligned} \max_{\mathbf{x},\boldsymbol{\tau}} \quad & y \\ \text{s.t.} \quad & \mathbb{P}_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij} \right) > y \right\} \geq 1 - \epsilon \end{aligned} \tag{2.9}$$

The goal of Eq. (2.9) is to maximize the worst-case score within the allowable risk threshold specified by ϵ (can be interpreted as guaranteeing that solution will be at least as good as y with probability greater than or equal to $1 - \epsilon$). When $\epsilon = 0$, the score is guaranteed to be at least y with probability one (absolute worst-case), and the chance-constrained formulation reduces to the robust formulation of Eq. (2.8). When $\epsilon = 0.5$, the chance-constrained formulation is equivalent to optimizing the median performance of the system (similar to optimizing the mean for symmetric distributions). Thus setting the risk value ϵ appropriately provides control over the conservatism of the solution within a consistent framework. The main drawback of the chance-constrained formulation is that it makes the optimization difficult to solve (analytically and computationally), especially given the extensive coupling between agents and tasks (double sum over distributions). Previous work has mainly considered linear or quadratic optimization with continuous variables [45, 66, 157], where, under special circumstances, optimal solutions and bounds can be found analytically. However, the task allocation formulation presented in Eq. (2.6) is a mixed-integer program, and these techniques cannot be easily extended to discrete optimization, especially given nonlinear and heterogeneous score functions with coupled distributions. Furthermore, these solution strategies are centralized and cannot be trivially extended to distributed environments. This thesis addresses these issues by proposing robust distributed planning strategies that can incorporate the stochastic metrics described in this section to improve the performance of multi-agent networked teams operating in uncertain and dynamic environments.

2.2 Solution Algorithms

Task allocation for autonomous multi-agent systems has been widely considered throughout the literature, and various researcher have addressed many aspects of the problem [19, 22, 52, 53, 112, 127, 148, 166, 184, 191, 198, 201], including traditional methods for solving Traveling Salesman Problems (TSPs) and Vehicle Routing Problems (VRPs) from the operations research (OR) and artificial intelligence (AI) communities [60, 213]. Exact optimization methods such as Branch and Bound, Branch and Cut, Constraint Satisfaction Problems (CSPs), and Dynamic Programming (DP) have been used to solve the problem to optimality, using standard software solvers such as CPLEX [102]. While guaranteed

to yield optimal results, these methods are computationally intensive, and the complexity associated with most combinatorial optimization problems typically make optimal solution techniques intractable. As a result, numerous approximation methods have been proposed to address these complexity issues.

An important observation is that, not only are optimal solution techniques computationally intractable for most problems of interest, they are also only optimal with respect to the objective function and underlying system models, which are often necessarily approximate, further motivating the use of approximation methods. Classical heuristic methods, such as constructive and two phase methods, have been used to solve large VRP problems relatively quickly [126], but these methods often generate solutions that are far from optimal. Different heuristic methods such as Tabu-Search [92], Cross-Entropy [64, 140, 215], Particle Swarm Optimization [62, 192], Genetic Algorithms [57, 72, 203] and Evolutionary Algorithms [158, 175] have also been proposed in recent years to solve these complex optimization problems. Although these approximations help to reduce the computation time as compared to exact methods, most of these algorithms are still computationally intractable for real-time replanning environments with complex constraints.

Several strategies have been considered in the literature to further reduce the computation time required to solve these problems. One approach is to reduce the problem size by limiting the duration (in time or plan length) using a receding horizon formulation [9, 52, 113]. A key challenge here, however, is to develop an effective and computationally efficient approximation of the "cost-to-go" from the terminal point of the plan, to avoid having an overly short-sighted planner. Other authors have identified several efficient mixed-integer linear programming formulations that dramatically reduce the number of variables and constraints in the problem, significantly alleviating the computational effort required. Although problem specific and applicable only in certain environments, these formulations cover a variety of UAV task assignment problems of practical interest [5, 6]. Other approaches have considered fast constant-factor approximate algorithms for multi-UAV assignment problems, which involve making mild assumptions to approximate the vehicle dynamics (e.g. constraints on turning radius, etc.) [179]. Finally, recent work has considered employing *learning* techniques to guide the MILP solution process [16]. The approach involves using machine learning to efficiently estimate the objective function values of the LP-relaxations to within an error bound, and to use that information to perform fast

inference at run time for new but similar relaxation problems that occur when the system dynamics and/or the environment changes slightly.

Although not the primary focus of this thesis, it is worth mentioning that planning for autonomous multi-agent teams has been considered within frameworks other than mixed-integer programming. In particular, the Markov Decision Process (MDP) framework has been widely studied in the literature for both deterministic and stochastic planning problems. The following is a very brief description of MDPs, however, it is interesting to note that the primary issues with mixed-integer multi-agent planning (e.g. computational tractability and scalability, agent modeling and representations, real-time dynamic planning, etc.) are major considerations within this framework as well. Markov Decision Processes provide a formal architecture for representing stochastic sequential decision making problems for multi-agent teams [49, 209]. Cooperative planners based on MDPs have shown tremendous versatility in modeling multi-agent systems [41, 96, 206, 217], and the associated solution algorithms, such as dynamic programming, have enabled near-optimal solutions in many application domains. As with mixed-integer programming however, a well-known issue is that MDPs and related dynamic programming techniques suffer from bad scalability [23], and quickly become intractable as the number of participating agents increases. As a result, many approximation algorithms have been developed to mitigate this computational challenge, including linear function approximation, Bellman residual minimization, least-squares temporal difference, and other approximate dynamic programming strategies [28, 48, 49, 75, 89, 124, 138, 196, 208, 210]. While many of these approaches have been successfully used to solve large multi-agent problems which would otherwise remain intractable [42, 206], most have fundamental challenges similar to mixed-integer programming methods, such as difficulty in selecting appropriate approximation architectures or absence of proper guidance in tuning algorithm parameters. For further details on MDPs and multi-agent MDPs the reader is referred to [42, 88, 171, 182].

Most of the approaches discussed in this section are centralized solution strategies, meaning that all of the planning relevant computation occurs in a single location. The following chapter discusses how to break this centralized assumption and highlights different strategies, issues, and challenges associated with distributing the optimization over multiple agents.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Distributed Planning

This chapter discusses various architectural decisions that must be addressed when implementing online planning algorithms for autonomous multi-agent teams. In particular, insights are provided on when centralized, distributed, and decentralized architectures can be used given the communication infrastructure and available computational resources of the multi-agent system. Different considerations and challenges associated with distributed planning are discussed, and algorithms that can be utilized within distributed planning frameworks are identified.

3.1 Distributed Planning Components

3.1.1 Planning Architectures

When planning for autonomous multi-agent networked teams, there are several planning architectures that can be considered depending on the mission scenario, available resources, operating environment, and communication infrastructure [105, 171, 201, 204]. This section formalizes definitions for three planning architectures that can be employed given different computational resources and communication environments: centralized, distributed, and decentralized. The differences between these architectures are highlighted with regards to how agents can handle consensus and cooperation in the assignment space, whether the underlying algorithms require synchronization or not, and how these choices affect the performance of algorithms within these respective environments[1].

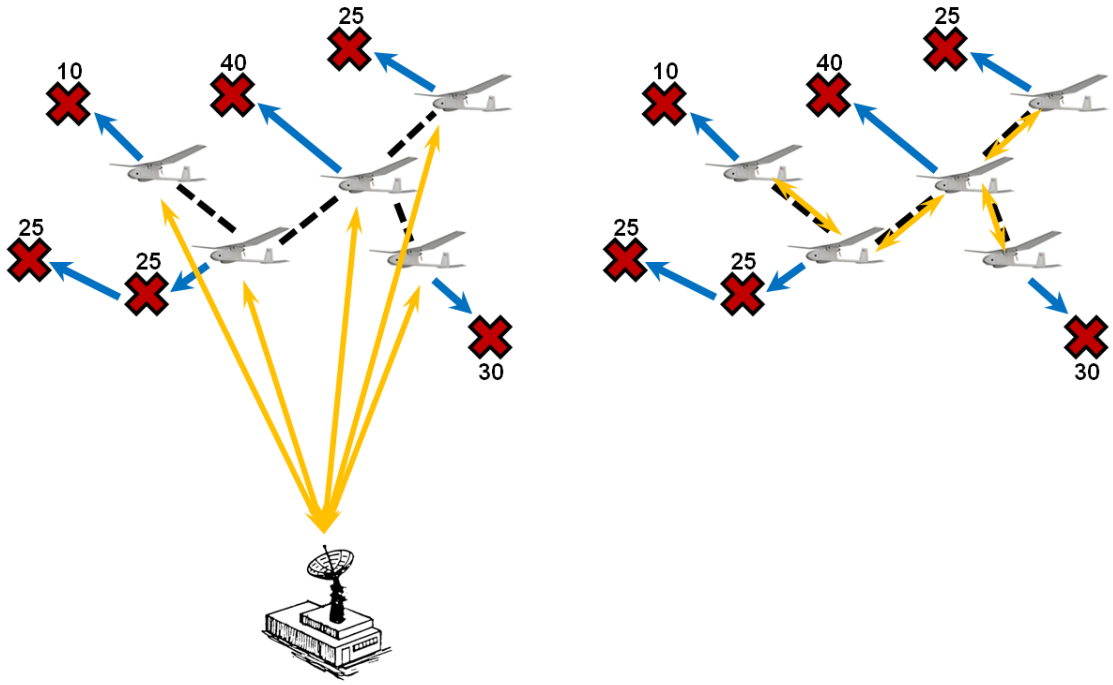
[1]For a more detailed description of centralized, distributed, and decentralized architectures, and the different issues associated with underlying planning algorithms within these environments, the reader is referred to [105, 171].

Centralized planning architectures refer to multi-agent algorithms that can run on a single machine, and where information exchange between different modules of the algorithm occurs through shared memory. Fast centralized algorithms are often implemented using parallelized computation structures that can take advantage of the large number of cores available in modern computer systems. Since all modules in the algorithm have instant access to the current global shared memory state, communication cost between modules is effectively negligible (communication cost includes both the additional resources required to facilitate communication and the associated delays introduced by explicit message passing). As a result, centralized planning architectures are often very good choices, even when planning for multi-agent teams. Figure 3-1(a) illustrates an example of planning for a networked team of UAVs using a centralized planning architecture. In some environments, however, enabling agents with more autonomy may be more suitable. For example, if planning parameters are changing rapidly (e.g. agent state, task parameters, environment variables), and large amounts of data have to be transmitted to the centralized solver's location, a centralized architecture may not be ideal. Remote processing of data to extract useful information prior to communicating with the base station may be more appropriate (referred to as distributed processing [204]), or the agents could converge locally on new assignments given the updated information without ever communicating with a centralized location (referred to as distributed problem solving [204]). Furthermore, the solution speed and quality of centralized solvers is limited by the accuracy of the communication and the rate at which information is received. Given slow, unreliable, or expensive communication channels, passing large amounts of (possibly irrelevant) data through the network may not be justifiable. If most of the information needed by each agent to create its cooperative assignments is obtained locally, much faster response times might be obtained by keeping all relevant planning computations local as well, and sharing only the results of these local computations with other agents.

Distributed planning architectures consist of algorithms that run as separate modules, where each of these distributed algorithmic pieces uses its own memory partition to store data associated with the planning process, and where relevant information is shared between the modules through reliable communication channels. This communication aspect introduces an additional layer of complexity over centralized algorithms, since there are communication costs and delays typically associated with sharing information. Distributed

algorithms typically rely on a *strong* communication infrastructure, meaning that each distributed node has knowledge of the existence of all other nodes it is able to communicate with, and assumes that messages between nodes will be sent and received reliably and with low latency. The main trade-off to consider when moving from a centralized to a distributed architecture is that the additional time and resources associated with message communication must be offset by the additional computation available by using multiple machines. For example, if the computational load of an optimization problem is too immense for a single (possibly multi-core) machine, distributed algorithms could be used to split the problem into modules over multiple machines communicating through reliable channels. For the multi-agent applications considered in this thesis, distributed algorithms are better suited than centralized algorithms, since they perform better when information is being acquired remotely (see Figure 3-1(b)). For example, in cooperative planning environments, this could involve an agent observing a local event, changing its own plan based on this newly observed information, and then reliably communicating the results to the rest of the distributed modules, as opposed to communicating the raw data associated with the local event. This local processing tends to dramatically reduce the overall communication load on the system while using a similar amount of computation as a centralized architecture. As mentioned before, distributed algorithms rely on stable communication channels and information sharing. If communication links are not sufficiently reliable, performance may degrade significantly, motivating the use of decentralized algorithms.

A *decentralized planning architecture* consists of independent agents planning autonomously for themselves in environments with unreliable and sporadic communication infrastructures, where there are no rigid constraints and guarantees placed on message delays, network connectivity, program execution rates, or message arrival reliability. In these types of environments, algorithms cannot rely on information being shared appropriately between the modules, thereby limiting the amount of coordination and cooperation achievable. As a result, these algorithms may be conservative and have lower performance than distributed architectures when communication conditions are actually favorable, but they are also more robust to drastic changes in the communication environment. Fully decentralized algorithms enable the sparsely connected agents with a high degree of autonomy, since they do not place rigid rules on how agents must interact. In addition, in weak communication environments, decentralized algorithms can allow large teams to interact efficiently, without bogging down



(a) Centralized Planning

(b) Distributed and Decentralized Planning

Figure 3-1: Illustration of centralized and distributed/decentralized planning for a networked team of UAVs performing target search and track tasks at different locations. In centralized planning, agents communicate with a ground station to receive plans. In distributed/decentralized architectures, agents can plan individually and perform consensus through local communication.

the entire network with restrictive message passing requirements.

Given the three planning architectures described above, choices can be made about how to coordinate the parallelized computation and communication amongst the different modules. In particular, consideration needs to be given about whether these interactions will happen synchronously or asynchronously and how that impacts the convergence and performance of the algorithms. Highly structured *synchronous* algorithms enforce constraints on when computation and communication can take place, whereas at the other extreme *asynchronous* algorithms provide extreme flexibility for each module to execute these at their own pace. Synchronization is typically used by most iterative algorithms, and involves restricting computations to occur only after certain event driven triggers, to ensure that the algorithm has a predictable state during program execution. During typical synchronous operation, the individual modules perform parallel computations, share state variables, and then wait until a certain trigger (e.g. heartbeat, scheduled time) before computing the next

iteration of the algorithm. The main benefit of synchronous planning is that guarantees can be made about the state of each agent, as well as allowing the algorithms to make assumptions about the information states of the other modules. This information can be leveraged to increase planning performance and rate of convergence. The drawback, however, is that in some environments it may take significant effort to enforce synchronous behavior (referred to as the synchronization penalty [29]). In distributed and decentralized algorithms, the computation triggers are often not local and must therefore be inferred through inter-module communication. In centralized approaches, the mechanisms required to enforce synchronization are fairly lightweight, but when information is being shared across a network, and modules must spend time waiting for messages to arrive from physically separated machines, this synchronization penalty can become severe. On the other hand, asynchronous computation can be used when the algorithmic modules are executed relatively independently of one another, such as in decentralized algorithms, where information is incorporated into the planning process whenever it is available instead of on a rigid schedule. There is fundamental trade-off, therefore, between enforcing synchronicity which allows the algorithm to make assumptions about algorithmic states, versus adopting a more flexible asynchronous architecture while taking a performance hit for losing information assumptions and state guarantees. Typically, in centralized and distributed environments, the synchronization penalty is weak enough to encourage the use of synchronous computation, while in decentralized approaches the synchronization penalty is much higher and asynchronous computation is often preferable.

3.1.2 Coordination Techniques

As mentioned in Chapter 2, in most general cases of interest, the score functions and constraints in the optimization are dependent upon the team decision variables. This produces nontrivial coupling between the agents when solving the optimization within a distributed or decentralized architecture. Since each agent's decisions depend upon the assignments and plans of the other agents in the team, the agents need to coordinate and reach agreement on consistent values of the decision variables to maximize team performance, or at least to ensure constraint satisfaction. In this section, three strategies are discussed to achieve this consistency: (1) *a priori* task space partitioning, where the task space is divided into disjoint sets and allocated amongst the agents prior to planning; (2) implicit coordination,

where agents perform situational awareness consensus and then plan independently; and (3) coordinated planning, where agents directly incorporate constraint feasibility into the planning process through the use of communication and consensus algorithms.

A priori task space partitioning effectively divides the task space into disjoint sets, and only allows each agent to select assignments from a subset of the overall task set. Given a higher-level task allocation or a human supervisor, it may be reasonable to use this method for small teams in relatively static environments, especially when roles and responsibilities of each agent are well defined in advance (e.g. specialized agents specifically designed for certain tasks). However, in environments with large numbers of relatively homogeneous agents, or in dynamic environments, the task partitioning problem can become very complex. Although this strategy enables rapid plan computation for each agent and ensures conflict-free team assignments, by creating this partition outside of the optimization the algorithm is placing artificial constraints on which allocations are available, which may result in arbitrarily poor performance.

The second strategy, implicit coordination, requires that agents perform consensus on *situational awareness* (SA) prior to running the planning algorithm. Situational awareness consensus refers to the process by which all agents agree on all variables relevant to the initial conditions of the task allocation problem (e.g. environment variables, agent states, task parameters, etc.). After this consensus process, the agents independently run centralized planners to obtain the global team assignment, and then select their portion of the plan to execute. The basic premise of this method is that, by running identical planners with consistent planning parameters, each agent will generate identical final allocations that satisfy constraint feasibility and maximize team performance. This method has been widely explored in the literature [9, 98, 151, 159, 162, 184–186, 211, 222], and its popularity stems from the fact that it is a relatively straightforward way to distribute a task allocation algorithm. One main benefit of using implicit coordination over task space partitioning is that, by not limiting the assignment space *a priori*, the algorithm can account for local information changes often encountered in dynamic environments (e.g. dynamic tasks, additional state information, updated model parameters, etc.). In the task space partitioning method, any changes in planning state usually require a full repartitioning of the task space, whereas implicit coordination can account for information changes through the situational awareness consensus protocol during replans, thus producing more relevant and higher per-

forming assignments. Furthermore, by solving the full centralized planner on each agent, implicit coordination can capture and exploit coupling and cooperation between agents well, and can therefore lead to good performance and cooperative behaviors in highly coupled environments. A drawback, however, is that the situational awareness consensus process which must be executed prior to the planning phase requires coming to consensus on all planning parameters, which may be time consuming and require large amounts of communication bandwidth [9]. This is especially cumbersome in robust planning environments where consensus must be performed not just on parameter values but on uncertainty models (e.g. distributions) as well. Another potential issue with implicit coordination algorithms is that planning parameters must be known precisely in order to guarantee that agents produce identical assignments, thus requiring that the consensus process be conducted until errors in situational awareness become very small. In fact, since the planning process does not explicitly guarantee constraint satisfaction, it is often the case that if the final estimates of the planning parameters do not converge to within arbitrarily tight bounds, the resulting team assignment may not be conflict free.

Finally, the third coordinated planning strategy involves performing *task consensus* by directly incorporating constraint feasibility into the planning process through the use of communication and consensus algorithms, thus guaranteeing conflict-free solutions. Since the communication effort is spent on achieving consistent values for the agent assignments rather than the situational awareness planning parameters, the bandwidth requirements are often lower, especially in scenarios with few inter-agent and inter-task constraints (the more coupled the task environment, the more communication effort it takes to explicitly ensure constraint feasibility). Furthermore, if the communication environment is not necessarily reliable such that it would be difficult to reach complete team-wide consistent situational awareness, these types of coordinated planning strategies are often preferred, to ensure that the team assignment remains conflict free even with varying situational awareness amongst the agents. For example, in many decentralized applications, the primary goal is achieving a conflict-free distribution of tasks that performs well, rather than requiring intense cooperation between agents to achieve the best possible assignment. Given that in these decentralized environments the communication links are often unreliable, especially across larger distances, broadcasting only information directly relevant to the assignment constraints may be preferable (e.g. the CBBA algorithm [58] described later in Section 4.1).

There is a trade-off between implicit coordination and assignment consensus, where the choice is motivated by the specific planning problem and the communication environment, and whether it is easier to converge on a consistent assignment vector or to converge on all other significant state variables (within arbitrarily tight bounds).

For the scenarios considered in this thesis, given the size and static nature of the assignment vector $\mathbf{x}$, the independence and simplification assumptions described in Chapter 2, and the relatively light constraints imposed on the system, it is typically easier to employ the third strategy to explicitly ensure conflict-free assignments through task consensus, rather than performing situational awareness consensus on all the dynamic and uncertain environment parameters. The CBBA algorithm and its variants discussed later in Chapter 4 use this third strategy to perform distributed planning. It is worth noting that the three coordination strategies discussed above are not mutually exclusive, and cooperative planning algorithms may use combinations of all three to improve system performance given the particular application and communication environment[2]. Ongoing research is addressing how to combine these three strategies to optimize performance and the interested reader is referred to the work of Johnson et al.[3] The next section provides details on consensus algorithms employed by distributed multi-agent teams to perform coordination between agents.

3.1.3 Consensus

A key component of distributed cooperative decision making involves performing *consensus* amongst agents, which is defined as reaching an agreement on quantities of interest, such as plans, situational awareness, or other desired parameters. Most distributed planning approaches employ *consensus algorithms*, which are sets of rules, or protocols, that determine how information is exchanged between agents to ensure that the team will convergence on the parameters of interest.

As a simple illustrative example, the following linear consensus protocol can be used to

[2]For example, Section 4.3 proposes an extension to CBBA to handle communication-limited environments, which involves combining (local) task space partitioning and task consensus to achieve conflict-free assignments in the presence of network disconnects.

[3]http://acl.mit.edu/members/johnson.htm

converge on a continuous parameter z,

$$\begin{aligned} \dot{x}_i(t) &= \sum_{j \in N_i} (x_j(t) - x_i(t)), \quad \forall i \\ x_i(0) &= z_i, \quad z_i \in \mathbb{R} \end{aligned} \tag{3.1}$$

where each agent i computes errors with its set of neighbors N_i and uses these to correct its parameter estimate [161]. Collectively the team dynamics for n agents can be written as an n^{th} order linear system,

$$\dot{\mathbf{x}}(t) = -\mathcal{L}\mathbf{x}(t) \tag{3.2}$$

where $\mathcal{L} = D - A$ is known as the graph Laplacian, which is computed using an adjacency matrix A describing connectivity between agents (the elements a_{ij} are 1 if j is a neighbor of i and 0 otherwise), and a degree matrix $D = \text{diag}(d_1, \ldots, d_n)$, with elements $d_i = \sum_{j=1}^{n} a_{ij}$ (number of connections for agent i). The maximum degree, denoted as $\Delta = \max_i d_i$, is useful in bounding the eigenvalues of $\mathcal{L}$, which for an undirected connected network can be ordered sequentially as

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \leq 2\Delta. \tag{3.3}$$

The eigenvalues of $\mathcal{L}$ can be used to predict convergence rates and stability properties of these linear consensus algorithms (in particular, λ_2 is related to speed of convergence and λ_n provides stability bounds in time-delayed networks [161]). As shown above, the nontrivial eigenvalues are all positive (all except λ_1), and since Eq. (3.2) describes a linear system, the consensus algorithm is globally asymptotically stable and converges exponentially to an equilibrium with rate given by λ_2 [161]. Furthermore, for the system described in Eq. (3.1), the algorithm is guaranteed to achieve a unique equilibrium, $\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i$, where $\bar{z}$ is the average of all the agents' initial values.

Recent research has explored the effects of more realistic mission environments on these types of linear consensus algorithms for multi-agent teams. Some examples include analyzing the impact of time-delayed messages and dynamic network topologies on convergence and stability properties of the consensus algorithms. The work in [161] shows that global exponential convergence guarantees can be extended to dynamic networks as long as the network remains connected at each time t. The agents are guaranteed to reach consensus

with convergence rate greater than or equal to $\lambda_2^\star = \min_t \lambda_2(G(t))$, where $\lambda_2(G(t))$ is the second eigenvalue of the Laplacian for the graph at time t, $G(t)$. Similar guarantees can be made for time-delayed networks, where messages are received after a delay τ instead of instantaneously. The system dynamics can be modified as follows,

$$\dot{\mathbf{x}}(t) = -\mathcal{L}\mathbf{x}(t-\tau) \tag{3.4}$$

and global exponential convergence guarantees are retained for delays within the range $\tau < \pi/2\lambda_n$. Note that convergence rates and robustness to time-delays can be improved by actively controlling the network structure (modifying G and $\mathcal{L}$), which is an active area of research [46, 103, 161, 184, 186].

Consensus algorithms have been applied to a wide variety of distributed decision making applications, ranging from flocking to rendezvous [20, 76, 103, 135, 161, 183, 184]. Most of these consensus algorithms are computationally inexpensive and guarantee convergence of team situational awareness, even over large, complex, and dynamic network topologies [98, 211, 222]. A common issue with classical consensus algorithms is that agents' observations are often treated with equal weight, whereas in reality some agents may have more precise information than others. Extending classical consensus algorithms to account for this uncertainty in local information, Kalman consensus approaches have been developed that approximate the inherent uncertainty in each agent's observations using Gaussian distributions [8, 185]. These algorithms produce consensus results that are more heavily influenced by agents with smaller covariance (therefore higher certainty) in their estimates. A limitation of Kalman consensus approaches is that Gaussian approximations may not be well-suited to model systems with arbitrary noise characteristics, and applying Kalman filter based consensus methods to the mean and covariance of other distributions can sometimes produce biased steady-state estimation results [82].

Other Bayesian decentralized data and sensor fusion methods have been explored to determine the best combined Bayesian parameter estimates given a set of observations [95, 139, 218]. A major challenge, however, is that these decentralized data fusion approaches require *channel filters* to handle common or repeated information in messages between neighboring nodes. These channel filters are difficult to design for arbitrary network structures, and generic channel filter algorithms have not been developed other than for simple

network structures (e.g. fully connected and tree networks), thus limiting the applicability of decentralized data fusion methods [95]. Recent work has addressed this issue by showing that, through a combination of traditional consensus-based communication protocols and decentralized data fusion information updates, scalable representative information fusion results can be achieved, without requiring complex channel filters or specific network topologies [82, 83, 160, 223]. In particular, the work in [160] utilized dynamic-average consensus filters to achieve an approximate distributed Kalman filter, while [223] implemented a linear consensus protocol on the parameters of the information form of the Kalman filter, permitting agents to execute a Bayesian fusion of normally-distributed random variables. However, as previously noted, these Kalman based methods are derived specifically for normally-distributed uncertainties [160, 223], and thus can produce biased results if the local distributions are non-Gaussian. Recent work has extended these combined filtering and data fusion approaches to allow networked agents to agree on the Bayesian fusion of their local uncertain estimates under a range of non-Gaussian distributions [83]. In particular, the approach exploits *conjugacy* of probability distributions [87], and can handle several different types of conjugate distributions including members of the exponential family (e.g. Dirichlet, gamma, and normal distributions) [82, 83]. The approach in [83] is termed *hyper-parameter consensus*, and has demonstrated flexibility in handling several realistic scenarios, including ongoing measurements and a broad range of network topologies, without the need for complex channel filters.

3.1.4 Distributed Performance Metrics

This section discusses several metrics that are useful in characterizing the performance of distributed algorithms. These metrics serve as a benchmark to assess the usefulness of different algorithms given the relative importance of each metric for the particular scenario at hand.

- *Score performance* measures the overall score obtained by the team given the objective function defined. It serves to characterize the importance of finding a solution that optimizes the global objective function, as opposed to optimizing other metrics such as convergence time. The relative importance of score performance is scenario dependent. For example, for scenarios with significant coupling between agents and

tasks, finding a good plan becomes difficult and typical distributed algorithm strategies might perform poorly. In these problems it may be desirable for an algorithm to have strong performance guarantees with respect to the optimal. On the other hand, in other scenarios it may be relatively easy to find near-optimal allocations, and other considerations might become the limiting factors.

- *Run time* specifically refers to how much computational time the entire algorithm takes to complete. This is typically considered a global measure for the entire team, as opposed to local convergence time as discussed below. Acceptable measures of run time can vary significantly depending upon the application at hand. For example, for offline solutions, acceptable run times are usually considered in terms of hours (or possibly days). On the other hand, for real-time dynamic systems such as those considered in this thesis, acceptable run times are usually on the order of seconds.

- *Convergence time* is a more flexible metric that can be used to describe the amount of time required to obtain a solution at the agent level or at the team level. In terms of global algorithmic convergence, run time and convergence time are equivalent. When referring to local agent convergence or partial system (component) convergence, convergence time measure the time required for the particular algorithmic piece to complete its computations. As intuition suggests, smaller convergence times are preferable, since fast convergence times allow for more rapid adaptation to dynamic changes in the environment. For highly dynamic and uncertain environments, replanning often is a useful tool, and thus convergence time is typically considered one of the most important metrics.

- *Reaction time*, which is closely related to convergence time, looks specifically at the turn around time between acquiring information and incorporating it into the plan cycle. In some less flexible algorithms, information cannot be incorporated into the planner mid-convergence, whereas other algorithms (such as decentralized ones) allow for the inclusion of new information at any time. During algorithm design, trade-offs between convergence time and score performance must be addressed to enable faster reaction times to local situational awareness changes.

- *Convergence detection* is typically a very difficult thing to quantify in distributed

and decentralized environments. During post processing given global information, it is usually trivial to detect when convergence occurred. However, at the local agent level, it may be difficult to determine if the full system has reached convergence (this is especially true in decentralized environments). As a result, different algorithms have employed different definitions of "convergence", including full global convergence, local module convergence, and partial system convergence (e.g. decisions about the next immediate tasks are consistent, but future decisions are still unsure) [105, 171]. The type of convergence criteria employed will impact what type of information is communicated as well as how much total communication is actually required by the system.

This section presented various architectural and performance considerations that need to be addressed when planning for autonomous multi-agent systems. For a more thorough review of distributed planning considerations the reader is referred to [105, 171]. The next section provides more details on the specific distributed problem formulation considered in this thesis, discusses some of the assumptions employed and outlines common solution approaches considered in the literature.

3.2 Distributed Planning Algorithms

3.2.1 Distributed Problem Formulation

Recall from Chapter 2 that the multi-agent task allocation formulation, with all the assumptions discussed in Section 2.1.3, can be written as

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\mathbf{p}_i(\mathbf{x}_i), \boldsymbol{\theta})\ x_{ij} \right) \\ \text{s.t.} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_i, \quad \forall i \in \mathcal{I} \\ & \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\ & x_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \end{aligned} \tag{3.5}$$

To solve this optimization in a distributed fashion, each agent can optimize its own plan, subject to the joint team constraints and coupling in score functions with other agents. For this particular problem statement, this implies that each agent $i \in \mathcal{I}$ must solve the following optimization,

$$\begin{aligned} \max_{\mathbf{x}_i} \quad & \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\mathbf{p}_i(\mathbf{x}_i), \boldsymbol{\theta})\ x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_i, \\ & \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\ & x_{ij} \in \{0,1\}, \quad \forall j \in \mathcal{J} \end{aligned} \tag{3.6}$$

where each agent i selects its best assignment vector $\mathbf{x}_i$. The main issue with this distributed formulation is the coupling between agents' decision vectors. In Eq. (3.6), this coupling comes in through the second set of constraints,

$$\sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J}$$

which specify whether an assignment is conflict free (i.e. at most one agent can be assigned to each task). Therefore, agent i's decision for task j, x_{ij}, depends upon other agents' decisions for task j. This coupling between agents becomes an issue in distributed environments and solution algorithms must carefully design consensus protocols to address the coupling. For example, if no other agent is assigned to task j (i.e. $x_{kj} = 0,\ \forall k \neq i$), then agent i is free to make its own independent decision for x_{ij}. If, however, one or more other agents are assigned to task j, then agent i must decide whether this task is infeasible for him, or whether he should "overrule" the other agents' decisions and add it anyway, informing the others that the task is infeasible for them instead. The CBBA algorithm [58], discussed later in Section 4.1, addresses this issue through an *auction algorithm*, where agents place bids on tasks, and the highest bidders get to keep their assignments. The consensus protocol of CBBA is designed specifically to address the joint constraint in Eq. (3.6), ensuring convergence to conflict-free assignments (given certain criteria on score functions, details provided later). The formulation presented in Eq. (3.6) does not include very severe coupling between agents,

however for more general scenarios, the more coupling there is (e.g. more joint constraints, coupled score functions) the harder it becomes to design adequate consensus protocols.

The time-varying version of Eq. (3.6) considered in this thesis, which corresponds to the full time-varying optimization presented earlier in Eq. (2.6), Chapter 2, can be written as follows,

$$\begin{aligned}
\max_{\mathbf{x}_i, \boldsymbol{\tau}_i} \quad & \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_i, \\
& \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\
& x_{ij} \in \{0,1\}, \quad \forall j \in \mathcal{J} \\
& \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall j \in \mathcal{J}
\end{aligned} \tag{3.7}$$

where each agent must optimize the additional decision variables $\boldsymbol{\tau}_i$ corresponding to the execution times of the assignments selected in $\mathbf{x}_i$. In this distributed time-varying problem statement, the only coupling between agents is again through the joint constraint specifying conflict-free assignments, and therefore the same deconfliction rules can be used in the consensus protocol. Of interest in this thesis are the robust variants of the distributed optimization presented in Eq. (3.7). As described in Section 2.1.4, robust metrics that can be used include the expected value metric, optimizing worst-case performance, and chance-constrained optimization which optimizes worst-case performance within an allowable risk threshold. Chapters 5 and 6 present the robust variants of this distributed problem formulation and provide details on how robust distributed optimization can be executed in uncertain and dynamic real-time environments. The next section describes how to design distributed algorithms and highlights different distributed approaches that have been considered in the literature.

3.2.2 Distributed Solution Strategies

As mentioned before, the main issue associated with distributed planning is deciding how to handle the coupling between individual agent problem statements. Therefore a key technical challenge involves designing appropriate consensus protocols to ensure constraint feasibility

and score maximization. The consensus protocol must specify what information agents must communicate with each other and define rules for agents to process received information and modify their plans accordingly. The specific design of the consensus protocol employed by a distributed algorithm has a huge impact on the convergence rate and the performance of the algorithm. Coupling can enter the optimization through agent score functions and through joint team constraints. Furthermore, agents' individual optimizations are based on each agent's local knowledge of the world and planning parameter values. Both coupling and varying situational awareness affect the performance of the global optimization since they directly impact agent score functions. The rules specified in the consensus protocol affect how agents share information and what decisions they make given new information. For example, in auction algorithms, when agents are outbid for assignments they typically drop those tasks and replan to select new ones. As a result, the deconfliction rules specified in the consensus protocol affect the rate of convergence of the algorithm. Furthermore, when designing consensus protocols consideration must be given to the communication environment and message passing requirements between agents. Given all these effects, it is often important to consider what performance and convergence guarantees can be made for different distributed algorithms.

Several approaches have been explored in the literature to solve distributed mixed-integer programming problems [51, 63, 148, 202]. Many of these methods often assume perfect communication links with infinite bandwidth in order to ensure that agents have the same situational awareness before planning. In the presence of inconsistencies in situational awareness, these distributed tasking algorithms are augmented with consensus algorithms to converge on a consistent state before performing the task allocation [9, 98, 151, 159, 162, 185, 186, 211, 222], a strategy described as implicit coordination in Section 3.1.2. Although these consensus algorithms guarantee convergence on information, they may take a significant amount of time and often require transmitting large amounts of data [7]. Alternately, distributed algorithms can be designed to perform consensus in the task space by sharing information about agent assignments rather than situational awareness. One class of planning algorithms in this category are *market-based or auction algorithms*, which are able to efficiently solve mixed-integer cooperative assignment problems in distributed and decentralized settings [68, 69]. Typically, these market-based approaches use an auction mechanism [27, 28], where each agent computes rewards associated with tasks or sets of

tasks, and then uses these reward values to place *bids* on the most desirable assignments. Auctions may be run via an auctioneer [90, 91, 145], where agents communicate their bids to a central location, and the auctioneer determines the winner for each task. These types of methods guarantee conflict-free solutions, since the auctioneer only selects one agent as the winner for each assignment, and are distributed since bids are calculated at spatially separated locations, however they do require a designated auctioneer (central location or specific agent) to resolve the conflicts. More flexible auction-based methods do not need to designate a single agent as the auctioneer, but utilize consensus protocols where the winner is decided based on a set of self-consistent rules [51, 58, 107]. Such methods have been shown to produce approximate solutions efficiently when reward functions satisfy a property called submodularity [3, 14, 106, 130, 194]. A particular algorithm of interest is the Consensus-Based Bundle Algorithm (CBBA) [58], a polynomial-time market-based approach that uses consensus to reach agreement on the cooperative assignment. The following chapter provides more details on CBBA and proposes key extensions that build upon the CBBA framework. Example applications involving multi-agent missions are presented, illustrating how these types of distributed auction algorithms can produce provably good approximate solutions in real-time distributed environments.

Although not the primary focus of this thesis, other major distributed multi-agent planning frameworks widely studied in the literature include Decentralized MDPs (Dec-MDPs) and Game Theory. Similar to mixed-integer programming methods, these frameworks suffer from many of the same issues and challenges as those described above, such as computational tractability and scalability, agent modeling and representations, communication and consensus design, and real-time dynamic planning. These frameworks are briefly described below and similarities to the distributed mixed-integer programming problem formulations are highlighted.

For most centralized multi-agent MDP formulations, all agents are assumed to have access to the global state. However, when agents are making decisions and observations locally this is generally an impractical assumption, since it would require each agent to communicate their observations to every other agent at every time step with zero communication cost (referred to as *free comm*). To address this issue, Decentralized MDP (Dec-MDP) formulations have been proposed that extend centralized multi-agent MDP problems to explicitly account for communication between agents [26], however, this additional layer increases

the dimensionality of the problem and thus the computational complexity[4]. Variants of Dec-MDPs include Dec-POMDPs to model uncertain environments, Dec-POMDP-COMs [93, 94] which explicitly model message passing as actions which incur costs, and Multi-Agent Team Decision Problems (MTDPs) and COM-MTDPs [176], which are very similar to Dec-POMDPs and Dec-POMDP-COMs[5]. These algorithms have all been shown to be NEXP-complete (nondeterministic exponential time), suffering from bad scalability as the number of agents and actions increases, motivating the development of approximate problem formulations that make assumptions on independence and communication between agents to reduce the computational complexity. Similar to mixed-integer programming algorithms, Dec-MDP approximation algorithms exploit the structure of the domain to make intelligent decisions about which assumptions to make given the scenario at hand (independence, feature-based representations, etc.), leading to trade-offs between optimality and computational tractability. The most notable of these approximation algorithms is the Transition Independent Decentralized MDP (TI-Dec-MDP) [21], which assumes that agents are independent collaborating entities connected through a global reward that is a function of all agents' states and actions (i.e. agents have independent transition and observation dynamics), thus reducing computational complexity. Other approximation algorithms include: the Decentralized Sparse-Interaction MDP (Dec-SIMDP) [149], which deals with computational limitations by separating the parts of the state space where agents need to cooperate from the parts where they can act independently; the Group-Aggregate Dec-MMDP [181], which uses features to compress other agents' state-action spaces to highlight the properties of interest for each agent (e.g. how many agents are in an area of interest, versus where is every agent in the environment); and the Auctioned POMDP [50], where each agent solves its own POMDP locally and communicates with other agents via an auction algorithm to achieve consensus on plans. Although these algorithms have demonstrated reduced computational complexity and real-time feasibility for large teams of cooperating agents, the approximation strategies involved are typically ad-hoc and problem dependent, and developing good approximation strategies for cooperating agents remains an active area of research.

[4]Dec-MDPs are NEXP (nondeterministic exponential time). As a reminder, the complexity hierarchy is given by $P \subseteq NP \subseteq PSPACE \subseteq EXP \subseteq NEXP$ denoting which classes of problems are subsets of other classes with regards to increasing computational complexity. For more details and discussion on complexity classes the reader is referred to [26, 163, 164].

[5]In fact, the MTDP, Dec-POMDP, COM-MTDP, and Dec-POMDP-COM have all been shown to be equivalent in terms of computational complexity and expressiveness of representation [199].

The field of game theory presents an alternate way of addressing the multi-agent planning problem by treating the interaction between agents as a game. The basic idea behind game theory is that agents are individual decision making entities that perform actions to maximize their own local utility based on knowledge of other agents and the environment. As such, game-theoretic frameworks are very similar to distributed mixed-integer programming strategies, and lend themselves naturally to solving autonomous task allocation problems in a distributed or decentralized fashion, motivating significant work in this area [12, 54, 85, 141, 142, 152, 155, 214]. Since in game theory agents make individual decisions about their own actions, these frameworks are useful for modeling noncooperative environments, however, enforcing cooperation is difficult because it involves ensuring that individual agent utility functions and incentives are aligned with the global mission goals. Therefore, the main challenge associated with game-theoretic cooperative planning strategies involves designing proper utility functions and negotiation strategies to ensure appropriate levels of coordination and collaboration between agents in order to maximize global mission performance, much like the consensus protocol design challenges described above for mixed-integer programming problems. Given that autonomous networked teams typically have limited bandwidth and communication constraints, it is desirable to find utility functions that keeps the amount of global information an agent requires to a minimum, while still aligning the agent's local utility with the global utility. A few localized utility functions proposed in the literature include Range-Restricted Utility, Equally Shared Utility and Wonderful Life Utility (see [12, 152, 214] for further details). Several multi-player learning algorithms have recently been developed that guarantee converge to stable feasible solutions (referred to as Pure-Strategy Nash Equilibria (PSNE) in the game-theoretic literature). Some of these include Action-Based Fictitious Play, Utility-Based Fictitious Play, Regret Matching (and variants) and Spatial Adaptive Play [12]. A primary issue with most of these distributed game theoretic algorithms is that while convergence to a PSNE is guaranteed, the resulting solution may often be far from optimal. To mitigate this problem it is necessary to carefully design negotiation strategies (similar to consensus protocols) such that the resulting algorithm converges to a near-optimal PSNE, a challenging task that has generated much interest in the research community [12, 152]. One algorithm that ensures convergence to a near-optimal PSNE with arbitrarily high probability is the Spatial Adaptive Play (SAP) algorithm [12], providing probabilistic guarantees that can be tuned

to obtain higher performance (at the expense of longer convergence times).

This section summarized many of the distributed planning algorithms considered in the literature. Of interest for this thesis is one particular algorithm, the Consensus-Based Bundle Algorithm (CBBA) [58], which provides guarantees on performance and convergence for autonomous multi-agent task allocation, and is well-suited to dynamic real-time planning environments. The next chapter describes CBBA, proposes key extensions that build upon the CBBA framework, and presents some example applications to illustrate how these distributed planning strategies can be used for real-time multi-agent mission planning.

Chapter 4

Consensus-Based Bundle Algorithm (CBBA) and Extensions

This chapter describes the Consensus-Based Bundle Algorithm (CBBA) developed by Choi et al. [58], and presents key extensions and variants. Section 4.1 describes the baseline CBBA algorithm proposed in [58], Section 4.2 discusses how to modify the bundle construction process to handle time-varying score functions within CBBA, and Sections 4.3 and 4.4 propose algorithms that build upon CBBA to enable mission execution in communication-limited environments. Section 4.3 presents a local distributed algorithm to ensure conflict-free solutions in the presence of network disconnects, and Section 4.4 proposes a cooperative distributed algorithm where agents can act as each other's relays to prevent network disconnects during task execution.

4.1 CBBA Algorithm Description

The Consensus-Based Bundle Algorithm (CBBA), originally developed by Choi, Brunet, and How [58], is a distributed auction algorithm that provides provably good approximate solutions for multi-agent multi-task allocation problems over random network structures. CBBA has been shown to work well in practice for various application, and can handle complex agent and task models within a real-time distributed framework, guaranteeing converge to conflict-free solutions despite possible inconsistencies in situational awareness between agents. CBBA has provable performance guarantees, and ensures solutions that achieve at least 50% optimality [58], although empirically its performance is shown to be

Algorithm 1 CBBA($\mathcal{I}, \mathcal{J}$)

Initialize $\{\mathcal{A}_i, \mathcal{C}_i\}, \quad \forall i \in \mathcal{I}$
while $\neg converged$ **do**
  $(\mathcal{A}_i, \mathcal{C}_i) \leftarrow$ CBBA-Bundle-Construction$(\mathcal{A}_i, \mathcal{C}_i, \mathcal{J}), \quad \forall i \in \mathcal{I}$
  $\mathcal{C}_i \leftarrow$ CBBA-Communicate$(\mathcal{C}_i, \mathcal{C}_{N_i}), \quad \forall i \in \mathcal{I}$
  $(\mathcal{A}_i, \mathcal{C}_i) \leftarrow$ CBBA-Bundle-Remove$(\mathcal{A}_i, \mathcal{C}_i), \quad \forall i \in \mathcal{I}$
  $converged \leftarrow \bigwedge_{i \in \mathcal{I}}$ Check-Convergence$(\mathcal{A}_i)$
end while
$\mathcal{A} \leftarrow \bigcup_{i \in \mathcal{I}} \mathcal{A}_i$
return $\mathcal{A}$

above 90% optimality [37]. The bidding process runs in polynomial time, demonstrating good scalability with increasing numbers of agents and tasks, making it well suited to real-time dynamic environments. This section describes the Consensus-Based Bundle Algorithm in detail, discussing the different algorithm phases and highlighting some key algorithmic subtleties.

The basic CBBA algorithm consists of iterations between two phases: a *bundle construction* phase where each agent greedily generates an ordered bundle of tasks, and a *task consensus* phase where conflicting assignments are identified and resolved through local communication between neighboring agents. These two phases are repeated until the algorithm reaches convergence. A summary of the overall Consensus-Based Bundle Algorithm is provided in Algorithm 1. To further explain the relevant details of CBBA, some notation will first be formalized. Each agent i keeps track of certain data structures associated with its own assignment status $\mathcal{A}_i$ or with its local knowledge of the current winning agents and winning bids $\mathcal{C}_i$. The data structures associated with agent i's assignment $\mathcal{A}_i$ include:

- A *bundle*, $\mathbf{b}_i \triangleq \{b_{i1}, \ldots, b_{i|\mathbf{b}_i|}\}$, which is a variable length set of tasks currently assigned to agent i, whose elements are defined by $b_{in} \in \mathcal{J}$ for $n = \{1, \ldots, |\mathbf{b}_i|\}$. The current length of the bundle is denoted by $|\mathbf{b}_i|$, which cannot exceed the maximum length L_i (see Eq. (3.6)), and an empty bundle is represented by $\mathbf{b}_i = \emptyset$ with $|\mathbf{b}_i| = 0$. The bundle is ordered chronologically with respect to when the tasks were added (i.e. task b_{in} was added before task $b_{i(n+1)}$), which enables the algorithm to keep track of which tasks are dependent on others.

- A *path*, $\mathbf{p}_i \triangleq \{p_{i1}, \ldots, p_{i|\mathbf{p}_i|}\}$, which is a set of tasks containing the same tasks as the bundle, but whose order is used to represent the order in which agent i will execute

the tasks in its bundle. The path is therefore the same length as the bundle, with elements $p_{in} \in \mathbf{b}_i$ for $n = \{1, \ldots, |\mathbf{p}_i|\}$, and is not permitted to be longer than L_i (i.e. $|\mathbf{p}_i| = |\mathbf{b}_i| \leq L_i$).

The data structures associated with agent i's local knowledge of the current winning agents and winning bids $\mathcal{C}_i$ include:

- A *winning agent list*, $\mathbf{z}_i \triangleq \{z_{i1}, \ldots, z_{iN_t}\}$, of size N_t, with elements $z_{ij} \in \{\mathcal{I} \cup \emptyset\}$ for $j = \{1, \ldots, N_t\}$, indicating who agent i believes is the current winner for task j. Specifically, the value in element z_{ij} is the index of the agent who is currently winning task j according to agent i's local information, and is $z_{ij} = \emptyset$ if agent i believes that there is no current winner.

- A *winning bid list*, $\mathbf{y}_i \triangleq \{y_{i1}, \ldots, y_{iN_t}\}$, also of size N_t, where the elements $y_{ij} \in [0, \infty)$ represent the corresponding winners' bids and take the value of 0 if there is no winner for the task.

- And finally, a vector of *communication timestamps*, $\mathbf{t}_i \triangleq \{t_{i1}, \ldots, t_{iN_a}\}$, of size N_a, where each element $t_{ik} \in [0, \infty)$ for $k = \{1, \ldots, N_a\}$ represents the timestamp of the last information update agent i received about agent k, either directly or through a neighboring agent.

The bundle construction process, described in detail in Section 4.1.1, involves agents independently optimizing their own assignments $\mathcal{A}_i$, where each agent locally computes task scores and uses these to place *bids* on desirable tasks (Algorithm 1, Line 3). An important point to note is that these bids are computed in a distributed fashion, using each agent's local score functions, local knowledge of other agents' assignments, and local values for planning parameters (i.e. local situational awareness). The bundle construction phase is then followed by a task consensus phase, described in Section 4.1.2, where all agents share their local knowledge of winning agents and winning bids $\mathcal{C}_i$ with their neighbors N_i, and process the newly received information to update their local knowledge and assignment vectors (Algorithm 1, Lines 4 and 5). The consensus protocol of CBBA specifies what information agents must share with neighboring agents, how each agent can locally merge information about winning agents and winning bids after communications have occurred, and how agent assignment vectors can be updated given the newly acquired information.

The algorithm iterates over these two phases until convergence is reached. Note that while lines 6 and 8 of Algorithm 1 summarize the global convergence and global assignment status of CBBA, each agent locally detects its own convergence status (see Section 4.1.2 for details), and each agent returns its own assignment $\mathcal{A}_i$ along with information about the overall team assignment (contained in $\mathcal{C}_i$). The following sections provide more details on the different phases of CBBA.

4.1.1 Bundle Construction Phase

The bundle construction phase of CBBA (Algorithm 1, Line 3), involves each agent independently solving Eq. (3.6) by selecting which tasks it would like to execute given the full list of available tasks $\mathcal{J} \triangleq \{1, \ldots, N_t\}$ and the current assignment vector $\mathbf{x}$ which includes other agents' decisions up to the current time (summarized in $\mathcal{C}_i$ as described above). Given spatially separated task locations, optimizing the assignment vector (finding $\mathbf{x}_i^\star$) implicitly involves solving a path optimization problem for every possible feasible vector $\mathbf{x}_i$. In other words, for any $\mathbf{x}_i$ considered, the agent must solve the following optimization

$$\max_{\mathbf{p}_i} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\mathbf{p}_i(\mathbf{x}_i), \boldsymbol{\theta})\ x_{ij}$$

to find the optimal path $\mathbf{p}_i^\star$ and thus the optimal score for that given assignment vector. This path optimization problem is similar to the Traveling Salesman Problem (TSP) which is known to be NP-hard [36, 213]. The agent must then consider several possible task combinations (i.e. all different feasible $\mathbf{x}_i$ vectors) to obtain its optimal assignment $\mathbf{x}_i^\star$. Therefore, finding an optimal assignment, even at the single agent level is considered an NP-hard problem which scales poorly as the number of tasks increases.

Several strategies have been proposed to address this tractability issue for single agent task optimization. The first involves limiting the plan horizon to only allow decisions over the next L_i tasks, similar to the maximum path length constraint of Eq. (3.6). The smaller L_i is the fewer combinations exist for $\mathbf{x}_i$ (and the fewer options exist for $\mathbf{p}_i$ as well)[1]. A lot of

[1] All possible combinations for $\mathbf{x}_i$ are given by 2^{N_t} (i.e. a task is either assigned or not assigned). Imposing a maximum path length constraint limits the available combinations to $\sum_{k=0}^{L_i} \binom{L_i}{k} < 2^{N_t}$, thus making the problem more tractable. Note that $\sum_{k=0}^{N_t} \binom{N_t}{k} = 2^{N_t}$.

algorithms proposed in the literature only consider single-task allocation (i.e. the case where $L_i = 1$) and then replan to assign more tasks. Although significantly easier and faster to solve, these approaches yield poor performance due to their myopic nature. CBBA considers situations where $L_i > 1$, thus assigning *bundles* of tasks to agents, where L_i can be regulated given the available computational resources. The next strategy is to design approximation algorithms to reduce the solution space in order to maintain computational tractability. This can be accomplished using two methods: (1) partitioning the task space into bundles of tasks, where agents can select to add entire bundles, and (2) incrementally building bundles by selecting one task at a time. The first method effectively reduces the cardinality of the search space by grouping tasks together (can be thought of as a new smaller problem with $2^{\tilde{N}_t}$ "mega-tasks", where the possible combinations are $2^{\tilde{N}_t} < 2^{N_t}$). Bidding on bundles of tasks has been explored significantly in the literature [10, 65, 165], and has the advantage of being able to capture coupling between tasks better than incrementally building bundles (e.g. clustered tasks, dependent tasks, etc). The main disadvantage, however, is that a decision must be made on how best to partition bundles, where enumerating all possible combinations is intractable. Incrementally building bundles, on the other hand, involves selecting the next best task in a greedy fashion (next highest scoring task). This greedy task selection approach reduces the search space to $\mathcal{O}(N_t L_i)$ making it very attractive from a computational standpoint, however, it often leads to suboptimal performance since it is myopic and fails to adequately capture coupling between dependent tasks. Furthermore, complications with this approach involve deciding how to assign individual scores to tasks given task coupling in the bundle (e.g. clustered tasks where task 2 might only be attractive because task 1 is already in the agent's bundle). One approach is to compute *marginal scores*, or the improvement in bundle score as a result of adding the new task to the bundle. There are a few issues with this approach given dependencies between tasks (for example, a task may only be attractive because the agent will be in the area after doing another task, and if the first task is dropped, the second one is not valuable anymore). CBBA uses this sequential greedy approach with marginal scores to create bundles of tasks and bids for each task, however, restrictions are made on the allowable score allocations to address these coupling issues. More details regarding the specific CBBA process are provided next, but for a thorough description of the more general issues and options associated with different bundle allocation strategies the reader is referred to [105].

Algorithm 2 CBBA-Bundle-Construction($\mathcal{A}_i, \mathcal{C}_i, \mathcal{J}$)

while $|\mathbf{p}_i| \leq L_i$ **do**
  for $j \in \mathcal{J} \setminus \mathbf{p}_i$ **do**
    $J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} = \max_{n_j} \sum_{j=1}^{N_t} \mathbf{c}_{ij}((\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta})\ x_{ij}$
    $\Delta J_{ij}(\mathbf{p}_i) = J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} - J_{\mathbf{p}_i}$
    $h_{ij} = \mathbb{I}(\Delta J_{ij}(\mathbf{p}_i) > y_{ij})$
  end for
  $j^\star = \underset{j \in \mathcal{J} \setminus \mathbf{p}_i}{\text{argmax}}\ \Delta J_{ij}(\mathbf{p}_i)\ h_{ij}$
  if $(\Delta J_{ij^\star}(\mathbf{p}_i)\ h_{ij^\star} > 0)$ **then**
    $\mathbf{b}_i \leftarrow (\mathbf{b}_i \oplus_{\text{end}} j^\star)$
    $\mathbf{p}_i \leftarrow (\mathbf{p}_i \oplus_{n_j^\star} j^\star)$
    $z_{ij^\star} \leftarrow i$
    $y_{ij^\star} \leftarrow \Delta J_{ij^\star}(\mathbf{p}_i)$
  else
    break
  end if
end while
$\mathcal{A}_i \leftarrow \{\mathbf{b}_i, \mathbf{p}_i\}$
$\mathcal{C}_i \leftarrow \{\mathbf{z}_i, \mathbf{y}_i, \mathbf{t}_i\}$
return $(\mathcal{A}_i, \mathcal{C}_i)$

The full bundle construction process used within the CBBA framework is summarized in Algorithm 2. During this bundle construction phase, each agent starts with an initial assignment $\mathcal{A}_i = \{\mathbf{b}_i, \mathbf{p}_i\}$ from the previous CBBA iteration (empty for the first iteration), and with initial values of the current winning agents and winning bids $\mathcal{C}_i = \{\mathbf{z}_i, \mathbf{y}_i, \mathbf{t}_i\}$. Recall that the score agent i obtains for a given path is,

$$J_{\mathbf{p}_i} = \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\mathbf{p}_i, \boldsymbol{\theta})\ x_{ij} \tag{4.1}$$

where $x_{ij} = 1$ for all tasks in the path. To determine which task to add to the bundle next, each agent executes the following process for every available task $j \in \mathcal{J} \setminus \mathbf{p}_i$ (i.e. $j \in \mathcal{J}$, $j \notin \mathbf{p}_i$). First, task j is "inserted"[2] into the path at all possible locations n_j to find the optimal position in the path,

$$J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} = \max_{n_j} \sum_{j=1}^{N_t} \mathbf{c}_{ij}((\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta})\ x_{ij} \tag{4.2}$$

[2]The notion of inserting task j into the path at location n_j involves shifting all path elements from n_j onwards by one and changing path element at location n_j to be task j (i.e $p_{i(n+1)} = p_{in}, \forall n \geq n_j$ and $p_{in_j} = j$).

where each new path is denoted $(\mathbf{p}_i \oplus_{n_j} j)$, with $\oplus_n$ signifying that task j is inserted at location n_j, and where $n_j^\star$ is the optimal location for task j that maximizes the above score (Algorithm 2, line 3). The marginal score for task j is then given by the increase in score as a result of adding task j,

$$\Delta J_{ij}(\mathbf{p}_i) = J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} - J_{\mathbf{p}_i} \tag{4.3}$$

as specified in Algorithm 2, line 4. Once the marginal scores for all possible tasks are computed ($\Delta J_{ij}(\mathbf{p}_i)$ for all $j \in \mathcal{J} \setminus \mathbf{p}_i$), the scores need to be checked against the winning bid list, $\mathbf{y}_i$, to see if any other agent has a higher bid for the task. The binary variable $h_{ij} = \mathbb{I}(\Delta J_{ij}(\mathbf{p}_i) > y_{ij})$ is defined, where $\mathbb{I}(\cdot)$ is an indicator function that equals 1 if the argument is true and 0 if it is false, so that $\Delta J_{ij}(\mathbf{p}_i)\ h_{ij}$ will be nonzero only for viable bids (Algorithm 2, line 5). The final step is to select the highest scoring task to add to the bundle (Algorithm 2, line 7),

$$j^\star = \underset{j \in \mathcal{J} \setminus \mathbf{p}_i}{\operatorname{argmax}}\ \Delta J_{ij}(\mathbf{p}_i)\ h_{ij} \tag{4.4}$$

If the bid for this best task is positive, the bundle, path, winning agents list, and winning bids list are then updated to include the new task,

$$\begin{aligned} \mathbf{b}_i &\leftarrow (\mathbf{b}_i \oplus_{\text{end}} j^\star) \\ \mathbf{p}_i &\leftarrow (\mathbf{p}_i \oplus_{n_{j^\star}} j^\star) \\ z_{ij^\star} &\leftarrow i \\ y_{ij^\star} &\leftarrow \Delta J_{ij^\star}(\mathbf{p}_i) \end{aligned}$$

The bundle building recursion continues until either the bundle is full (the limit L_i is reached), or no tasks with positive score can be added for which the agent is not outbid by some other agent (i.e. $\Delta J_{ij}(\mathbf{p}_i)\ h_{ij} \leq 0$ for all $j \in \mathcal{J} \setminus \mathbf{p}_i$). Algorithm 2 provides a full description of the bundle construction process.

4.1.2 Task Consensus Phase

Once agents have built their bundles of desired tasks they need to communicate with each other to resolve conflicting assignments amongst the team. Each agent shares its winning agent list and winning bid list with neighboring agents, and this new information, along with timestamp data, is then passed through a decision table (see [58], Table 1 for details) that provides all of the conflict resolution logic to merge local bid information (CBBA-COMMUNICATE function, Algorithm 1, Line 4). For further details on this consensus process the reader is referred to [58], but in general, the consensus logic favours higher valued and more recent bids.

After merging information from neighboring agents about the winning agents and corresponding winning bids, each agent can determine if it has been outbid for any task in its bundle. Since the bundle building recursion, described in the previous section, depends at each iteration upon the tasks in the bundle up to that point, if an agent is outbid for a task, it must release it and all subsequent tasks from its bundle[3] (CBBA-BUNDLE-REMOVE function, Algorithm 1, Line 5). If the subsequent tasks are not released, then the marginal scores computed for those tasks would not be accurate leading to a degradation in performance. If the consensus phase has occurred more than twice the network diameter times without any change in bid information, then the algorithm has converged and terminates (Algorithm 1, Line 6); if not, each agent re-enters the bundle building phase and the algorithm continues. It should be noted that these conflict resolution rules specified by the CBBA consensus protocol explicitly address the conflict-free constraint of Eq. (3.6).

The worst-case complexity of the bundle construction phase of CBBA is $\mathcal{O}(N_t L_i)$ per iteration, and CBBA is guaranteed to converge within $\max\{N_t, L_i N_a\} D$ iterations, where D is the network diameter (always less than N_a). Thus, CBBA has polynomial-time convergence guarantees and scales well with the size of the network and/or the number of tasks, making it well suited to real-time applications. More detailed discussions regarding the consensus phase and the interaction between the two CBBA phases can be found in [58] and [105].

[3] A task in the bundle is only dependent on tasks that are located in the bundle prior to it, and is therefore independent of all tasks added after it. Keeping track of dependencies in this way has the effect of adding stability to the consensus space, since the number of dependencies between tasks is reduced and therefore convergence rates can increase.

4.1.3 Diminishing Marginal Gains

One requirement that is fundamental to all the convergence and performance guarantees of CBBA, is that the score functions used must satisfy a property called Diminishing Marginal Gains (DMG) . This DMG condition is a subset of the well-studied submodularity condition, and was recognized in the original description of CBBA [58], and further studied in recent work [106]. The DMG condition can be formalized as,

$$\Delta J_{ij}(\mathbf{p}_i) \geq \Delta J_{ij}(\mathbf{p}_i \oplus_{n_k^\star} k), \quad \forall j,k \in \mathcal{J} \setminus \mathbf{p}_i,\ j \neq k, \quad \forall i \in \mathcal{I} \tag{4.5}$$

Intuitively, the condition states that the score for any agent i and task j cannot increase as more things are added to the bundle (e.g. if a task was unattractive before, its appeal cannot suddenly increase as a result of a longer bundle). Given score functions that satisfy DMG, CBBA is guaranteed to converge to a conflict-free solution, however, if this condition is not met it could lead to cycling of assignments between the agents and the algorithm would fail to converge.

To further illustrate this cycling behavior consider the following simple scenario with 2 agents and 2 tasks. In the first case, summarized in Example 1, assume that score functions satisfy DMG. During bundle construction, assume Agent 1 computes a score $y_{11} = S$ for Task 1 and a score $y_{12} < S$ for Task 2, and therefore adds Task 1 to its bundle first, $\{(1, S)\}$, where the syntax denotes (task ID, task score). The agent then considers Task 2, and since the score functions satisfy the DMG condition of Eq. (4.5), y_{12} is necessarily less than y_{11} and will therefore be denoted as $y_{12} = S - \epsilon$ with $\epsilon > 0$. Task 2 is added to the bundle to give $\{(1, S), (2, S - \epsilon)\}$. Similarly, assume that Agent 2 builds its bundle, where Task 2 is preferred over Task 1, and the computed scores yield the following bundle: $\{(2, S), (1, S - \epsilon)\}$. During the consensus phase, Agent 1 realizes it has been outbid for Task 2 and drops the task from its bundle. Similarly, Agent 2 realizes it has been outbid for Task 1 and drops it from its bundle. The agents then return to the bundle construction phase, but since no more tasks can be added, a consistent assignment has been reached and the algorithm terminates.

When nonsubmodular score functions are used instead, the algorithm may result in cycles and convergence is not guaranteed. To illustrate this effect in the above example, consider the bundle construction for Agent 1 where Task 1 is added with score S as before,

Example 1: Task allocation with score functions that satisfy DMG

`Iteration 1`

Agent 1: $\{(1, S), (2, S - \epsilon)\}$

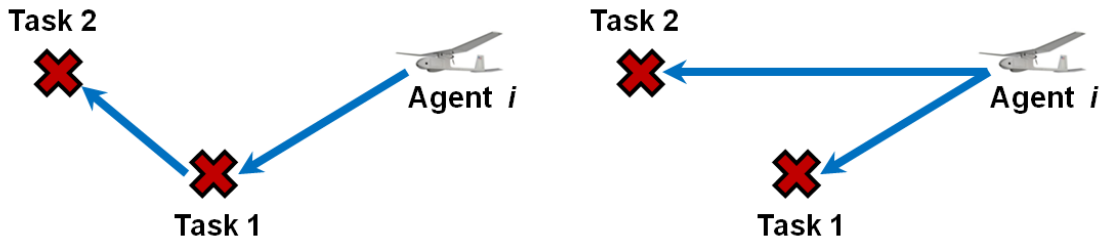
Agent 2: $\{(2, S), (1, S - \epsilon)\}$

`Iteration 2`

Agent 1: $\{(1, S)\}$

Agent 2: $\{(2, S)\}$

$\{(1, S)\}$. Task 2 is now considered, but since score functions need not satisfy DMG, the score for Task 2 as a result of adding Task 1 to the bundle could increase, $y_{12} > S$. This new nonsubmodular score is denoted as $y_{12} = S + \epsilon$ with $\epsilon > 0$, and Agent 1's bundle becomes, $\{(1, S), (2, S + \epsilon)\}$. Similarly, Agent 2 could construct a bundle $\{(2, S), (1, S + \epsilon)\}$ using a similar process. During the consensus phase, Agent 1 realizes it has been outbid for Task 1 and therefore must drop both Task 1 and Task 2 (since Task 2 depended on Task 1). Likewise, Agent 2 realizes it has been outbid for Task 2 and therefore drops both tasks from its bundle. The agents then enter the bundle construction phase, but since both agents still think that the other agent's bids are higher, no new tasks are added to either agents' bundles. The agents perform consensus again and each realizes that the other agent has dropped all its tasks. The bundle construction phase is entered for a 3^{rd} time, and since there are no valid bids for either agent the construction process is unconstrained and the result is the same as for Iteration 1. This process, summarized in Example 2, leads to a cycle that would continue forever, and the algorithm therefore would never converge to a consistent assignment.



(a) True Agent Path

(b) Heuristic Path Approximation

Figure 4-1: Example UAV mission with 1 agent and 2 tasks.

The above example shows the importance of satisfying the DMG condition, however,

Example 2: Task allocation with nonsubmodular score functions

`Iteration 1`

Agent 1: $\{(1, S), (2, S + \epsilon)\}$

Agent 2: $\{(2, S), (1, S + \epsilon)\}$

`Iteration 2`

Agent 1: $\{\}$

Agent 2: $\{\}$

`Iteration 3`

Agent 1: $\{(1, S), (2, S + \epsilon)\}$

Agent 2: $\{(2, S), (1, S + \epsilon)\}$

for many scenarios of interest this could be a limiting requirement. For example, consider a multi-agent multi-task UAV mission, such as that depicted in Figure 2-1, where agents must perform target search and track tasks which involve traveling to and servicing tasks at different locations. The planning parameters in this scenario include elements such as task locations, task rewards, fuel penalties, etc. The total objective function for each agent can be written as,

$$J_i = \left(\sum_{j=1}^{N_t} \mathbf{R}_j \, x_{ij} \right) - f_i \, d_i(\mathbf{p}_i) \tag{4.6}$$

where $\mathbf{R}_j$ is a fixed reward for executing task j, f_i is the fuel cost per unit distance, and $d_i(\mathbf{p}_i)$ represents the total distance traveled by agent i given path $\mathbf{p}_i$. This score function, which is a very naturally occurring example, is not submodular due to the fuel penalty component, since some tasks may look more attractive when more tasks are added to the path if the travel distances become shorter. As an example, consider the scenario in Figure 4-1(a) with task rewards $\mathbf{R}_j = R, \ \forall j$. In the first iteration of the bundle construction process, the scores obtained for Tasks 1 and 2 are $y_{i1} = (R - f_i \, d_{i1})$ and $y_{i2} = (R - f_i \, d_{i2})$ respectively, where d_{i1} and d_{i2} represent the distances from the agent's initial position to the task locations. Since $d_{i1} < d_{i2}$, the score for Task 1 is higher and therefore this task is added to the path. In the second iteration, the score for Task 2 is computed using the

marginal score calculation $\Delta J_{ij}(\mathbf{p}_i) = J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} - J_{\mathbf{p}_i}$ which yields,

$$\begin{aligned} y_{i2} &= (2R - f_i\,(d_{i1} + d_{12})) - (R - f_i\,d_{i1}) \\ &= (R - f_i\,d_{12}) \end{aligned}$$

Since $d_{12} < d_{i2}$, the new score for Task 2, $y_{i2} = (R - f_i\,d_{12})$, is higher than it was during the first iteration (since most of the fuel cost is subsumed into the score of the first task), which breaks the DMG condition specified in Eq. (4.5). This type of score function using fuel penalties is a very natural example arising in many situations (e.g. clusters of tasks, traveling salesman problem), and therefore the DMG condition can be quite limiting. In early work [174], we proposed a submodular heuristic version of the score function specified in Eq. (4.6), where the true path distance was approximated using a distance heuristic,

$$\hat{J}_i = \sum_{j=1}^{N_t} \left(\mathbf{R}_j - f_i\,\hat{d}_{ij}\right)\,x_{ij} \tag{4.7}$$

In Eq. (4.6), $\hat{d}_{ij}$ is the distance from the agent's *initial* position to task j instead of the true distance (see Figure 4-1(b)), but since $\hat{d}_{ij}$ remains constant for each task j regardless of which tasks are in the path before or after it, the score for any task j cannot increase as the path becomes longer and therefore the DMG condition of Eq. (4.5) is satisfied.

As shown in the above example, implementing CBBA for real-world multi-agent problems involves designing heuristic score functions such as Eq. (4.7) that can approximate the true score functions while still satisfying the diminishing marginal gains property. In general, using heuristic approximations within the planner leads to suboptimal performance with respect to using the true score functions. Therefore choosing a good heuristic function is a key design step which may not be easy given the scenario at hand. This issue is further aggravated in stochastic planning situations, where stochastic metrics and numerical approximations make heuristic DMG satisfying score functions difficult to design. However, in several scenarios of interest involving multi-agent teams, good heuristics score functions can be created and CBBA can be successfully employed, especially when the suboptimality arising from using approximate score functions can be mitigated by replanning.

As a last major note, recent work by Johnson et al. [106] proposed an algorithmic extension to embed the DMG condition into the algorithmic framework itself, enabling

the use of CBBA with arbitrary score functions. This extension alleviates the burden of having to design heuristic score functions, and is explained fully in Section 5.2.2, especially with regards to how it enables the distributed stochastic planning framework described in Chapters 5 and 6.

4.2 CBBA with Time-Varying Score Functions

Of interest for this thesis are dynamic environments with time-varying rewards, as described in Chapter 2 and Section 3.2.1. In these environments agents must optimize the time-varying score function,

$$\max_{\mathbf{x}_i, \boldsymbol{\tau}_i} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij}$$

where the decision variables include the proposed task execution times $\tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}$ corresponding to the task assignment decision variables x_{ij}. In this section, we propose an extension to the baseline CBBA algorithm to handle these time-varying score functions in the bundle construction process, describing the additional variables and algorithmic steps (see publication [174]).

4.2.1 Bundle Construction with Time-Varying Score Functions

To enable the use of time-varying score functions, the CBBA bundle construction process can be modified to explicitly include optimization of task execution times. The agent assignment $\mathcal{A}_i$ can be augmented to include an additional data structure $\boldsymbol{\tau}_i$, representing a vector of times $\boldsymbol{\tau}_i \triangleq \{\tau_{i1}, \ldots, \tau_{i|\boldsymbol{\tau}_i|}\}$, whose elements are defined by $\tau_{in} \in [0, \infty)$ for $n = \{1, \ldots, |\boldsymbol{\tau}_i|\}$, denoting the times at which agent i will execute the tasks in its path. The process to compute task scores is modified as follows and is summarized in Algorithm 3. For all tasks $j \in \mathcal{J} \setminus \mathbf{p}_i$, each task j is inserted into the path at all possible locations n_j to find the optimal position in the path. For each location n_j, this step involves finding the optimal times $\boldsymbol{\tau}_i^\star$ for all tasks in the new path (Algorithm 3, line 4),

$$\boldsymbol{\tau}_i^\star = \underset{\boldsymbol{\tau}_i}{\operatorname{argmax}} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta})\ x_{ij} \tag{4.8}$$

and repeating the process described in Eq. (4.8) for all locations n_j, giving the following expression for the cost (Algorithm 3, line 6),

$$J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} = \max_{n_j} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^\star(\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta})\ x_{ij} \tag{4.9}$$

The marginal score for task j is then given by the increase in score as a result of adding task j (Algorithm 3, line 7),

$$\Delta J_{ij}(\mathbf{p}_i) = J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} - J_{\mathbf{p}_i} \tag{4.10}$$

and the optimal task to add is given by,

$$j^\star = \underset{j \in \mathcal{J} \setminus \mathbf{p}_i}{\operatorname{argmax}}\ \Delta J_{ij}(\mathbf{p}_i)\ h_{ij} \tag{4.11}$$

as before (Algorithm 3, line 10). The bundle, path, times, winning agents list, and winning bids list are then updated to include the new task,

$$\begin{aligned}
\mathbf{b}_i &\leftarrow (\mathbf{b}_i \oplus_{\text{end}} j^\star) \\
\mathbf{p}_i &\leftarrow (\mathbf{p}_i \oplus_{n_{j^\star}} j^\star) \\
\boldsymbol{\tau}_i &\leftarrow (\boldsymbol{\tau}_i \oplus_{n_{j^\star}} \tau_{ij^\star}^\star(\mathbf{p}_i \oplus_{n_{j^\star}} j^\star)) \\
z_{ij^\star} &\leftarrow i \\
y_{ij^\star} &\leftarrow \Delta J_{ij^\star}(\mathbf{p}_i)
\end{aligned}$$

The recursion continues until no tasks can be added anymore. The bundle construction process for time-varying CBBA is summarized in Algorithm 3. The marginal score calculations including time optimization satisfy the DMG condition of Eq. (4.5), since adding more tasks to the path has the effect of further constraining the time optimization process of Eq. (4.8). In other words, as more tasks are added to the path, there are less possible options for the task execution times $\boldsymbol{\tau}_i$ subject to agent and environment dynamics, and thus the values $\boldsymbol{\tau}_i^\star$ computed in Eq. (4.8) are the result of a more constrained optimization problem. Therefore, as long as the original (non-time-varying) score functions satisfy DMG, the extra step of optimizing task execution times does not affect the requirement of

Algorithm 3 TIME-VARYING-CBBA-BUNDLE-CONSTRUCTION$(\mathcal{A}_i, \mathcal{C}_i, \mathcal{J})$

while $|\mathbf{p}_i| \leq L_i$ **do**
  for $j \in \mathcal{J} \setminus \mathbf{p}_i$ **do**
    for $n_j \in \{1, \ldots, (|\mathbf{p}_i| + 1)\}$ **do**
      $\boldsymbol{\tau}_i^\star = \underset{\boldsymbol{\tau}_i}{\text{argmax}} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta})\ x_{ij}$
    end for
    $J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} = \max_{n_j} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^\star(\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta})\ x_{ij}$
    $\Delta J_{ij}(\mathbf{p}_i) = J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} - J_{\mathbf{p}_i}$
    $h_{ij} = \mathbb{I}(\Delta J_{ij}(\mathbf{p}_i) > y_{ij})$
  end for
  $j^\star = \underset{j \in \mathcal{J} \setminus \mathbf{p}_i}{\text{argmax}}\ \Delta J_{ij}(\mathbf{p}_i)\ h_{ij}$
  if $(\Delta J_{ij^\star}(\mathbf{p}_i)\ h_{ij^\star} > 0)$ **then**
    $\mathbf{b}_i \leftarrow (\mathbf{b}_i \oplus_{\text{end}} j^\star)$
    $\mathbf{p}_i \leftarrow (\mathbf{p}_i \oplus_{n_j^\star} j^\star)$
    $\boldsymbol{\tau}_i \leftarrow (\boldsymbol{\tau}_i \oplus_{n_j^\star} \tau_{ij^\star}^\star(\mathbf{p}_i \oplus_{n_j^\star} j^\star))$
    $z_{ij^\star} \leftarrow i$
    $y_{ij^\star} \leftarrow \Delta J_{ij^\star}(\mathbf{p}_i)$
  else
    break
  end if
end while
$\mathcal{A}_i \leftarrow \{\mathbf{b}_i, \mathbf{p}_i, \boldsymbol{\tau}_i\}$
$\mathcal{C}_i \leftarrow \{\mathbf{z}_i, \mathbf{y}_i, \mathbf{t}_i\}$
return $(\mathcal{A}_i, \mathcal{C}_i)$

Eq. (4.5).

Even though the CBBA with Time-Varying Score Functions algorithm operates in continuous time, the specific task execution time optimizations can exploit a few properties associated with this problem formulation to maintain computational tractability. The first property is that there is a causal dependence between tasks in the path, therefore early tasks are not affected by later tasks in the path. Leveraging this causal relationship, the optimization of task execution times in Eq. (4.8) involves optimizing these task times sequentially, where the optimal task time for the first task in the path becomes a constant when optimizing the second task in the path, etc. Using this sequential process greatly reduces the search space associated with the optimization and, due to the causal dependence between tasks, produces the same results as optimizing all the task execution times simultaneously. Now that the optimization of Eq. (4.8) involves optimizing each continuous decision variable separately, and since each variable τ_{ij} only affects the corresponding task score $\mathbf{c}_{ij}$, we can leverage knowledge of the particular task score functions and employ

fast line search algorithms to find the best τ_{ij} that optimizes $\mathbf{c}_{ij}$ subject to the temporal constraints specified by tasks earlier in the path. For certain types of score functions (e.g. unimodal functions, functions of the forms specified in Figure 2-2), efficient line search algorithms such as gradient descent and Newton's method can be used to find the optimal task times, enabling real-time optimization of Eq. (4.8) within the CBBA with Time-Varying Score Functions framework (Algorithm 3, line 4).

It is important to note that during the iterative bundle construction process, the task execution time optimization of Eq. (4.8) involves re-optimizing all the times for tasks in the path after location n_j at every step to appropriately represent the impact of adding task j to the path. Even after leveraging the properties described above when optimizing Eq. (4.8), this re-optimization of task execution times is typically a computationally intensive step which slows down the convergence rate of the algorithm significantly (as compared to the original CBBA). A constraint that can be imposed to reduce this computational effort and speed up convergence, is to restrict new tasks to be inserted into the path only if they do not impact the proposed times for the tasks already in the path. This constrained optimization involves computing the time for the new task j only, without having to re-compute the times of the tasks later in the path, thus reducing the required computations. The optimization can be written as,

$$\begin{aligned} \tau_{ij}^{\star}(\mathbf{p}_i \oplus_{n_j} j) = \quad & \underset{\tau_{ij}\in[0,\infty)}{\operatorname{argmax}} \ \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta}) \\ \text{subject to:} \quad & \tau_{ik}^{\star}(\mathbf{p}_i \oplus_{n_j} j) = \tau_{ik}^{\star}(\mathbf{p}_i), \quad \forall k \in \mathbf{p}_i \end{aligned} \tag{4.12}$$

where the constraints state that the insertion of the new task j into path $\mathbf{p}_i$ cannot impact the current times (and corresponding scores) for the tasks already in the path [174]. Using this constraint, the marginal score is simply the score for the new task j, since the scores for other tasks are not subject to change, therefore

$$\Delta J_{ij}(\mathbf{p}_i) = \mathbf{c}_{ij}(\tau_{ij}^{\star}(\mathbf{p}_i \oplus_{n_j^{\star}} j), \boldsymbol{\theta})$$

which reduces computation time significantly. The DMG property is again satisfied for this modified problem, as explained below. To ensure the DMG condition of Eq. (4.5) the

following is required,

$$\mathbf{c}_{ij}(\tau_{ij}^{\star}(\mathbf{p}_i \oplus_{n_j^{\star}} j), \boldsymbol{\theta}) \geq \mathbf{c}_{ij}(\tau_{ij}^{\star}(\mathbf{p}_i' \oplus_{n_j^{\star}} j), \boldsymbol{\theta}), \quad \forall j \in \mathcal{J} \setminus \mathbf{p}_i$$

where the score for a task not currently in the path can only decrease as more tasks are added to the path, and where $\mathbf{p}_i' = (\mathbf{p}_i \oplus m)$ for any task $m \in \mathcal{J} \setminus \mathbf{p}_i$, $m \neq j$. Since the calculation of the best arrival time for task j when the current path is $\mathbf{p}_i'$ instead of $\mathbf{p}_i$ is given by,

$$\begin{aligned} \tau_{ij}^{\star}(\mathbf{p}_i' \oplus_{n_j} j) = \quad & \underset{\tau_{ij} \in [0,\infty)}{\operatorname{argmax}} \ \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i' \oplus_{n_j} j), \boldsymbol{\theta}) \\ \text{subject to:} \quad & \tau_{ik}^{\star}(\mathbf{p}_i' \oplus_{n_j} j) = \tau_{ik}^{\star}(\mathbf{p}_i'), \quad \forall k \in \mathbf{p}_i' \end{aligned}$$

the constraints can be rewritten recursively as the following set of constraints,

$$\begin{aligned} &\tau_{ik}(\mathbf{p}_i \oplus m \oplus_{n_j} j) = \tau_{ik}^{\star}(\mathbf{p}_i \oplus m) = \tau_{ik}^{\star}(\mathbf{p}_i), \quad \forall k \in \mathbf{p}_i \\ &\tau_{im}(\mathbf{p}_i \oplus m \oplus_{n_j} j) = \tau_{im}^{\star}(\mathbf{p}_i \oplus m) \end{aligned}$$

Therefore, calculation of $\tau_{ij}^{\star}(\mathbf{p}_i' \oplus_{n_j} j)$ involves solving an optimization with the same objective function but an additional constraint. Thus, the optimal objective value for this optimization cannot be greater than that for $\tau_{ij}^{\star}(\mathbf{p}_i \oplus_{n_j} j)$, in other words $\mathbf{c}_{ij}(\tau_{ij}^{\star}(\mathbf{p}_i \oplus_{n_j^{\star}} j), \boldsymbol{\theta}) \geq \mathbf{c}_{ij}(\tau_{ij}^{\star}(\mathbf{p}_i' \oplus_{n_j^{\star}} j), \boldsymbol{\theta})$, which means the DMG property is satisfied.

This work is significant since most algorithms that can handle time-windows in the literature are centralized and/or computationally intensive. For example, recent game theoretic approaches have attracted significant interest as distributed task allocation algorithms for assigning tasks to agents in time-varying environments [12, 54, 141, 142], however, these approaches rely on discretizing time, severely increasing the computational complexity of the problem. In contrast, CBBA with Time-Varying Score Functions does not require time discretization, and, for certain score function types as described above (e.g. unimodal), the algorithm is able to address the optimization of these additional decision variable in polynomial-time, ensuring both spatial and temporal coordination amongst the agents, while still preserving the robust convergence properties of the original algorithm.

4.2.2 Example Applications

To show the real-time applicability of CBBA with time-varying score functions, consider a dynamic planning scenario involving a multi-agent multi-task UAV/UGV mission, such as that depicted in Figure 2-1, where agents must perform *time-critical* tasks which involve traveling to and servicing tasks at different locations and during different times (see Figure 2-2). This type of scenario could include rescue operations, where victims must be found and attended to in a timely manner, or even time-critical services within urban environments such as pizza delivery or taxi routing. This section provides details on the implementation of CBBA for multi-agent networked teams operating in dynamic environments.

Simulation Results

A simulation was created to demonstrated the performance of CBBA with Time-Varying Score Functions when planning for multi-agent networked teams operating in time-critical mission scenarios. For these types of time-varying missions, the planning parameters $\boldsymbol{\theta}$ associated with agents and tasks could include elements such as task locations, task reward, task service times, task time-windows of validity, agent positions, agent travel velocities, fuel penalties, etc. In this thesis, we consider time-varying task rewards of the following form,

$$\mathbf{R}_{ij}(\tau_{ij}) = \begin{cases} \mathbf{R}_j \; e^{-\lambda_j \Delta\tau_{ij}}, & t_{j_{\text{start}}} \leq \tau_{ij} \leq t_{j_{\text{end}}} \\ 0, & \text{otherwise} \end{cases}$$

where $t_{j_{start}}$ is the first time the task becomes available, the task time-window $[t_{j_{start}}, t_{j_{end}}]$ represents the period of time in which the task must be completed, τ_{ij} is the time at which agent i finishes executing task j, and $\Delta\tau_{ij} = \max\{0, \tau_{ij} - (t_{j_{\text{start}}} + t_{j_{\text{duration}}})\}$ represents the time in excess of the expected task completion time. The exponential decay represents the time-critical nature of the task, where the discount factor λ_j is used to reduce the nominal reward $\mathbf{R}_j$ according to the delay $\Delta\tau_{ij}$. The total objective function for each agent can be written as,

$$J_i = \sum_{j=1}^{N_t} \mathbf{R}_{ij}(\tau_{ij}) \; x_{ij} - f_i \; d_i(\mathbf{p}_i) \tag{4.13}$$

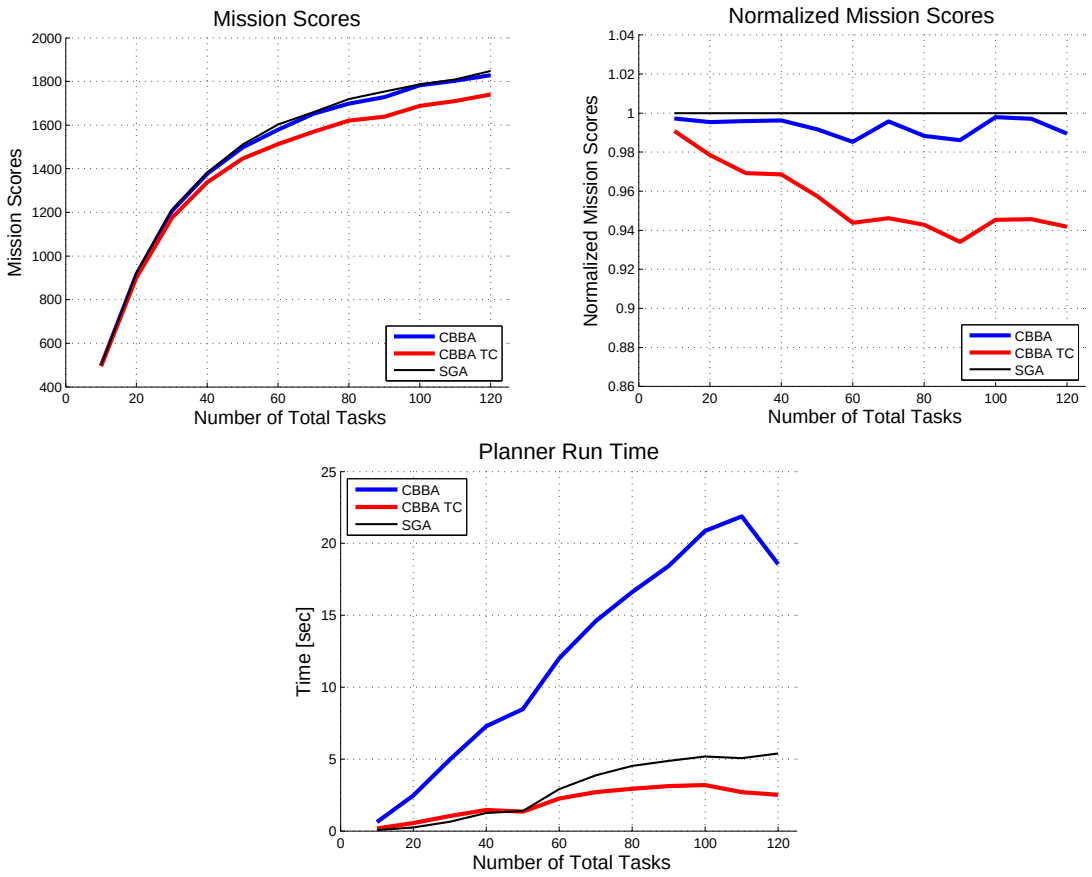


Figure 4-2: Monte Carlo simulation results for a 6 agent networked team performing time-critical missions. The simulations compare three planning algorithms: the CBBA with Time-Varying Score Functions algorithm proposed in this section (CBBA); CBBA with Time-Varying Score Functions using time constraints during sequential task selection (CBBA TC), as explained in Eq. (4.12); and a centralized sequential greedy algorithm (SGA). Figure (a) shows the achieved mission scores, (b) shows the mission scores normalized against the sequential greedy results, and (c) shows the planner run time as a function of the number of available tasks in the environment.

where $d_i(\mathbf{p}_i)$ represents the distance traveled by agent i given path $\mathbf{p}_i$ and f_i is the fuel cost per unit distance as described in the example of Section 4.1.3. Since this score function is not submodular due to the fuel penalty component, as explained in Section 4.1.3, a heuristic approximation that satisfies DMG can be employed instead,

$$\hat{J}_i = \sum_{j=1}^{N_t} \left(\mathbf{R}_{ij}(\tau_{ij}) - f_i \, \hat{d}_{ij} \right) \, x_{ij} \tag{4.14}$$

where $\hat{d}_{ij}$ is an approximation of the true distance which satisfies DMG (since $\hat{d}_{ij}$ is constant for each j regardless of which tasks are in the path). This approximate submodular score

function was used within the CBBA with Time-Varying Score Functions algorithm in simulation to approximate the agents' path scores while still satisfying the DMG requirement. Figure 4-2 shows Monte Carlo simulation results validating the performance of CBBA with Time-Varying Score Functions for a 6 agent networked team performing time-critical missions. The simulations compare three planning algorithms: the CBBA with Time-Varying Score Functions algorithm proposed in this section (CBBA); CBBA with Time-Varying Score Functions using time constraints during sequential task selection (CBBA TC), where tasks can only be added if they do not impact the other tasks already in the bundle as explained in Eq. (4.12); and a centralized sequential greedy optimization algorithm (SGA) which uses the true score function of Eq. (4.13) within the optimization (since DMG does not affect centralized algorithms). Figure 4-2(a) shows that the mission scores achieved using the distributed CBBA with Time-Varying Score Functions algorithm are similar to those obtained using a centralized sequential greedy optimization algorithm (SGA), validating the distributed approach and the submodular approximate score function. Figure 4-2(b) shows the mission scores normalized against the sequential greedy results, where CBBA with Time-Varying score functions achieves performance that is within 98% of that obtained using the centralized SGA approach. As shown in Figure 4-2(c), the planner run time required for the CBBA with Time-Varying Score Functions algorithm is higher than that of the centralized sequential greedy algorithm, since the algorithm requires more iterations to resolve conflicts amongst agents. The reduction in run time which occurs between 100 and 120 tasks is associated with the fact that agents have more choices and are therefore less likely to conflict, leading to fewer iterations of CBBA. In general, the distributed CBBA run time is a function of the number of tasks explored by each agent, the number of conflicts between agents (and thus iterations of CBBA), and the amount of time required to compute each path score within each agent's bundle optimization process. As a disclaimer, all computations were done in single threads, programmed in MATLAB, on an Alienware computer with an Intel Core i7 processor and 12 GB RAM, and these results show the total computation time for all agents combined. In practical real-time implementations, the computation would be distributed/parallelized over several computers and written in a more efficient language than MATLAB (e.g. C++), therefore the true computation time would be reduced by at least a factor of N (since each agent would compute its own plans), and would possibly be even faster given a more efficient programming language. Of course,

the distributed implementation would also have to account for communication speed between agents, which, depending on the application at hand, may introduce further delays (see [105] for a detailed analysis on message passing requirements of CBBA). As mentioned in the previous section, the time optimization step of Eq. (4.8) requires re-optimizing all the times for tasks in the path after location n_j to appropriately represent the impact of adding task j to the path, which can be quite computationally intensive. The constraint specified in Eq. (4.12) reduces this computational effort by only optimizing the task execution time of the new task j. As shown in Figure 4-2, the CBBA with Time-Varying Score Functions algorithm using this time constraint (CBBA TC) is able to significantly reduce the computation time required by the algorithm leading to much lower run times (Figure 4-2(c)), and the performance achieved by imposing this artificial constraint is still very close to that of the unconstrained CBBA (Figures 4-2(a)-(b)) and is within 93% of the centralized sequential greedy performance. The algorithms developed later in Sections 4.3 and 4.4 have higher computation requirements than the baseline CBBA with Time-Varying Score Functions algorithm, and thus can really benefit from the computational savings associated with this time constraint to enable real-time performance for networked teams.

Real-Time Replanning Architecture

In order to ensure that the task allocation remains relevant in a dynamically changing environment it is necessary to replan in real-time. Replanning at a fast enough rate ensures that vehicle states and network topologies are up to date, new tasks are accounted for and older or irrelevant tasks are pruned, and that the impact of discrepancies between the agent models in the planner and the actual agent behavior is minimized given the deterministic planning parameters. We have embedded the CBBA with Time-Varying Score Functions algorithm described in this section within a real-time dynamic planning framework for heterogeneous agents, illustrated in Figure 4-3. The overall architecture is comprised of a mission control center that manages a dynamic task list, the CBBA task allocation algorithm to coordinate planning for the team, models of the agents and the network structure that are updated in real time, and vehicle managers, actuators and sensors that enable agents to interact with the environment. The mission control center supplies an initial set of tasks with time-windows of validity to the agents, in addition to creating new pop-up tasks at a specified rate. The distributed CBBA planner receives the dynamic list of tasks and

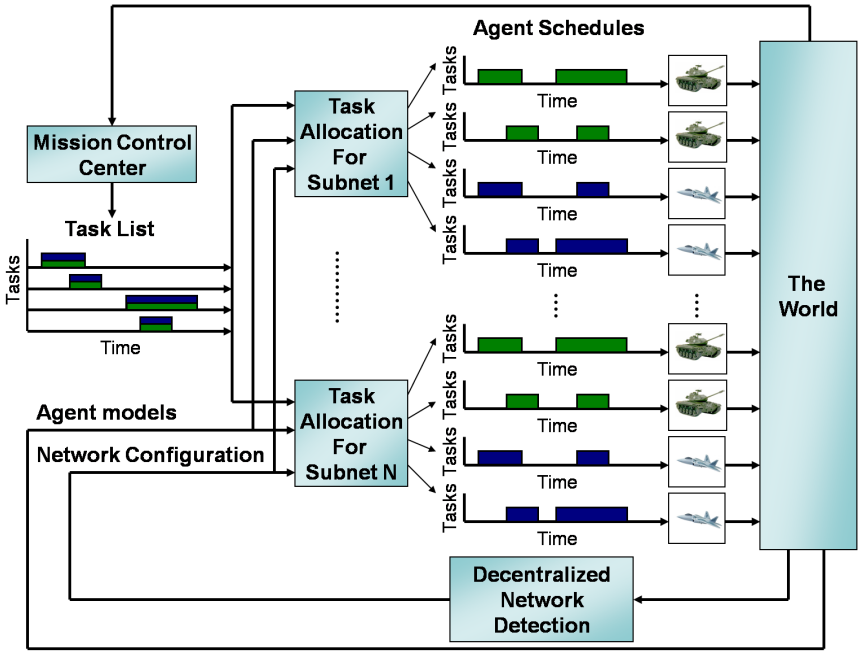


Figure 4-3: Real-time distributed task allocation architecture for a heterogeneous networked team.

allocates them to the agents. The network detection algorithms use the position of the vehicles to determine the network graph and/or the set of subnetworks at any given time. And the vehicle managers consist of finite state machines that enable the execution of tasks for each agent. Given the latest agent models, network configuration information, and current task list, the planning architecture can allocate the tasks to the respective agents over some planning horizon, thereby creating schedules for each of the heterogeneous agents.

This real-time replanning architecture was used to validate the performance of the planning algorithms for heterogeneous networked UAV/UGV teams performing dynamic ISR missions, both in simulation and in experimental flight tests at MIT's Real-time indoor Autonomous Vehicle test ENvironment (MIT RAVEN) [99]. This indoor flight facility is equipped with motion-capture systems which yield accurate, high-bandwidth position and attitude data for all tracked vehicles within the flight volume. Flight experiments were conducted for heterogeneous teams of agents including quadrotor air vehicles, helicopter UAVs, and various ground vehicles, demonstrating the real-time applicability of the approach [174]. Figure 4-4 shows a snapshot of the simulation interface showing the agent paths and schedules over the course of the mission. The display on the left shows the agents and their proposed paths and the tasks along with a countdown to their expiry time. The

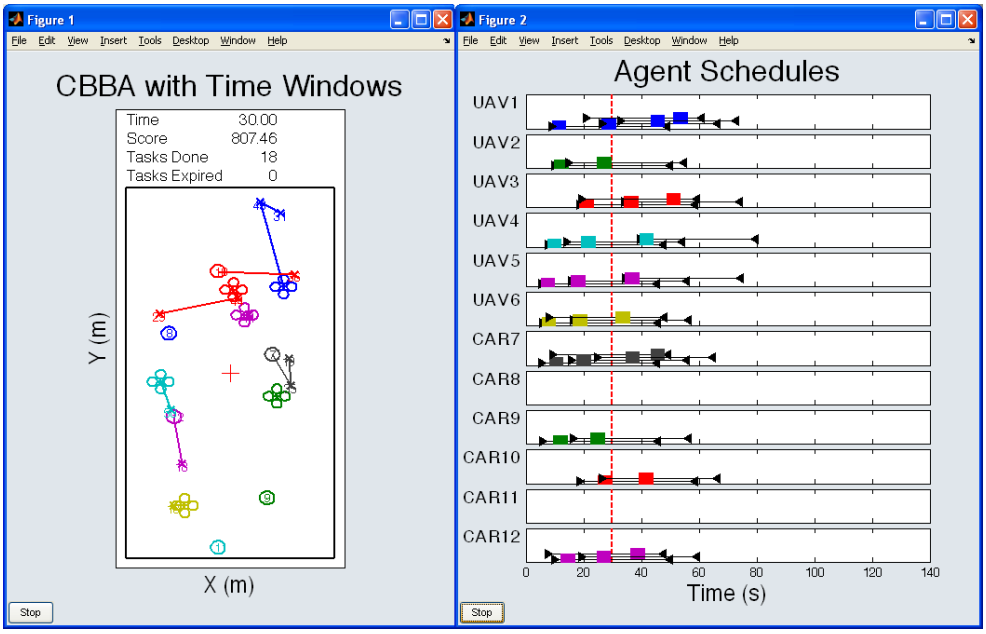


Figure 4-4: Simulation showing 12 agents (6 UAVs & 6 UGVs) bidding on and accomplishing a dynamic set of tasks.

Figure 4-5: Real-time mission planning for a heterogeneous networked team using the CBBA planning framework (Aerospace Controls Lab, MIT).

right display shows the agent schedules, including the past history and the proposed future schedule (on either side of the time line). The time-windows of validity for the tasks are shown (in black) along with the actual time that the agent executed the task (colored block). Figure 4-5 shows an example mission using this CBBA planning framework in a real-time flight experiment at the MIT Aerospace Controls Lab[4].

4.3 Distributed Planning with Network Disconnects

An issue associated with distributed planning using CBBA is that the planning algorithm will only converge if the agents maintain network connectivity. The network graph can change over time, but must be connected in order for agents to perform consensus. If the network becomes disconnected, agents will not know about other agents' bids outside of their sub-network, nor will they be able to communicate with these other agents in order to execute consensus. In this situation the planner may not converge and multiple agents from different sub-networks might bid on the same tasks leading to conflicting assignments. This section describes these communication challenges and proposes strategies for handling network disconnects.

4.3.1 Dynamic Network Handling Protocols

For missions involving multi-vehicle teams operating in communication-limited environments, the network structure is often dynamic. As agents move throughout the environment performing tasks, communication links between them can be dynamically created and destroyed, leading to varying network topologies and potentially disconnected networks. For example, if vehicles need to be within a certain distance of each other in order to communicate (*communication radius*), but if there exist some tasks such that a vehicle is forced to travel outside of this communication radius, then the vehicle must lose connectivity with its neighbors in order to accomplish these tasks. In these situations, CBBA fails to converge, since the vehicle is not able to communicate its current winning bids to the other agents, and thus the next round of replanning may assign that agent's tasks to other agents. This conflicting situation is undesirable since sending multiple agents to do the same tasks leads to unnecessary fuel consumption. Furthermore, it is assumed that when vehicles get within

[4]An online video demonstrating the CBBA planning framework is available at: http://acl.mit.edu/projects/cbba.html

communication distance of each other they will be able to resolve the assignment conflict, but if the planner replan rate is not fast enough, they may not be able to deconflict in time, possibly leading to collisions. It is necessary, therefore, to have a method to ensure that task assignments remain conflict-free in the presence of network disconnects.

One strategy that ensures conflict-free assignments involves using task space partitioning (see Section 3.1.2), where the task space is divided into disjoint sets and allocated amongst the sub-networks prior to planning. As discussed in Section 3.1.2, for several scenarios of interest the task partitioning problem can be very complex, and by creating a task partition outside of the optimization, the algorithm places artificial constraints on which allocations are available resulting in arbitrarily poor performance. Furthermore, doing task space partitioning at a centralized location (e.g. ground station) can be cumbersome, and involves having the centralized system partition the task space, and communicate the partition to the corresponding agents in the respective sub-networks, where each sub-network can then run distributed local planners. An alternate strategy is to have the agents locally partition the task space themselves. This can be accomplished by only allowing bids on tasks that are currently carried by agents in their specific sub-networks (including tasks won and new tasks that agents know about locally). This strategy ensures conflict-free assignments if the tasks each agent carries are unique, since viable tasks are only shared locally within each sub-network. Therefore, this local adjustment of the available task lists guarantees conflict-free team assignments in the presence of network disconnects, without requiring the intense communication and computational overhead associated with a ground station performing centralized *a priori* task space partitioning at every replan iteration.

In this work, three different methods of handling varying network topologies are compared. The first involves the default CBBA behavior, where all agents know about and are free to bid on any task in the entire task list (No task list adjustment). If the network is disconnected, the task allocation algorithm will run locally within each sub-network, and the global allocation may contain conflicting assignments between agents in different sub-networks. The second strategy requires that the mission control center perform task space partitioning, where the tasks are uniquely distributed amongst the agents by assigning each task only to the closest compatible agent (Central task list adjustment). Agents can then run the task allocation algorithm locally within their sub-network over the set of tasks that are assigned to agents in that sub-network to re-optimize the allocation if a better assign-

ment is achievable. This approach guarantees conflict-free assignments but requires the mission control center to redistribute the entire task list amongst the agents every time a replan is required, which for realistic missions would involve significant communication and computational overhead, limiting the real-time performance of the team. The third strategy considered is the local distributed algorithm proposed in this thesis, and involves replanning locally within each sub-network over the set of tasks that are currently in the paths and in the new task lists for all agents within that sub-network (Local task list adjustment). To ensure that the new task lists are unique between agents, the mission control center notifies only the closest compatible agent every time a new task is created. Since each agent's paths and new task lists are unique, running CBBA locally within each sub-network will lead to conflict-free assignments for the entire team. This local strategy has the additional benefit that each sub-network can decide to replan on their own schedule, as opposed to the centralized strategy that synchronizes replans between the sub-networks by distributing the task lists to all sub-networks at the same time. This can reduce computational overhead if certain sub-networks do not need to replan (i.e. nothing has really changed).

Both the central and local task list adjustment methods require that the mission control center maintain updated agent and task information, however, the local adjustment method requires that the mission control center communicate with the closest agent once per new task, whereas the central adjustment method requires communication messages to redistribute *all* the current tasks amongst the agents *every* time a replan is required. Thus the local adjustment method significantly reduces the amount of communication overhead required by the mission control center, while still ensuring deconflicted task assignments given disconnected agents. The next section describes experiments comparing these three different methods.

4.3.2 Example Applications

Simulation Results

The scenario used to test the different task adjustment approaches described above involved a team of 12 heterogeneous agents (6 UAVs and 6 ground robots). The simulation was initialized with 12 UAV tasks with random start times, 40 sec time windows and 5 sec durations. Once the UAV tasks were started a secondary ground robot rescue task was

created for each UAV task. Additional pop-up UAV tasks were created at 5 sec intervals and the task allocation algorithm replanned every 2 sec. The simulation consisted of a mission control center, a network detector, the distributed planning algorithms, and local agent simulations (see Figure 4-3). The network detector used the vehicle positions and a communication radius parameter to determine if two vehicles were able to communicate and returned a list of subnetworks. The local agent simulations implemented models of the vehicles to execute the tasks in each agent's path. The mission control center maintained the list of tasks by creating pop-up tasks and pruning completed tasks from the list, in addition to implementing the responsibilities for the different task adjustment methods described in the previous section. The overall mission score was obtained by adding the individual scores for the agents using the score function described in Section 4.2.2.

Using this simulation infrastructure as a testbed, the three methods described in the previous section were implemented, and Monte Carlo simulations of 200 iterations were executed to compare the mission performance for these three approaches under different communication radii. Figure 4-6 shows the overall mission scores, the number of completed tasks and the team fuel consumption as a function of the communication radius normalized by the maximum distance of the theater of operation. The results show that for all three methods the mission score increases as the communication radius increases, since agent coordination improves with communication. With a normalized communication radius of about 0.3 and higher and with a team of 12 agents, the network remains connected in most cases and all three methods yield similar performance. With less agents this communication radius threshold would be higher, since for a given communication radius, it is more likely that the network would lose connectivity with fewer agents. The baseline case (no adjustment) is seen to have the lowest score and highest fuel consumption, especially at low communication radii. This is because without task list adjustments there will be many assignment conflicts between different subnetworks, resulting in unnecessary fuel usage from having multiple agents attempt to perform the same tasks as well as a lower number of overall completed tasks (since agents are busy traveling to tasks that they will never accomplish). As the connectivity decreases and the number of subnetworks increases this problem becomes worse. With task list adjustments the mission performance greatly improves as seen in the results for both the central and local task list adjustment methods. Since the task allocation is guaranteed to be conflict-free over the entire team there is no

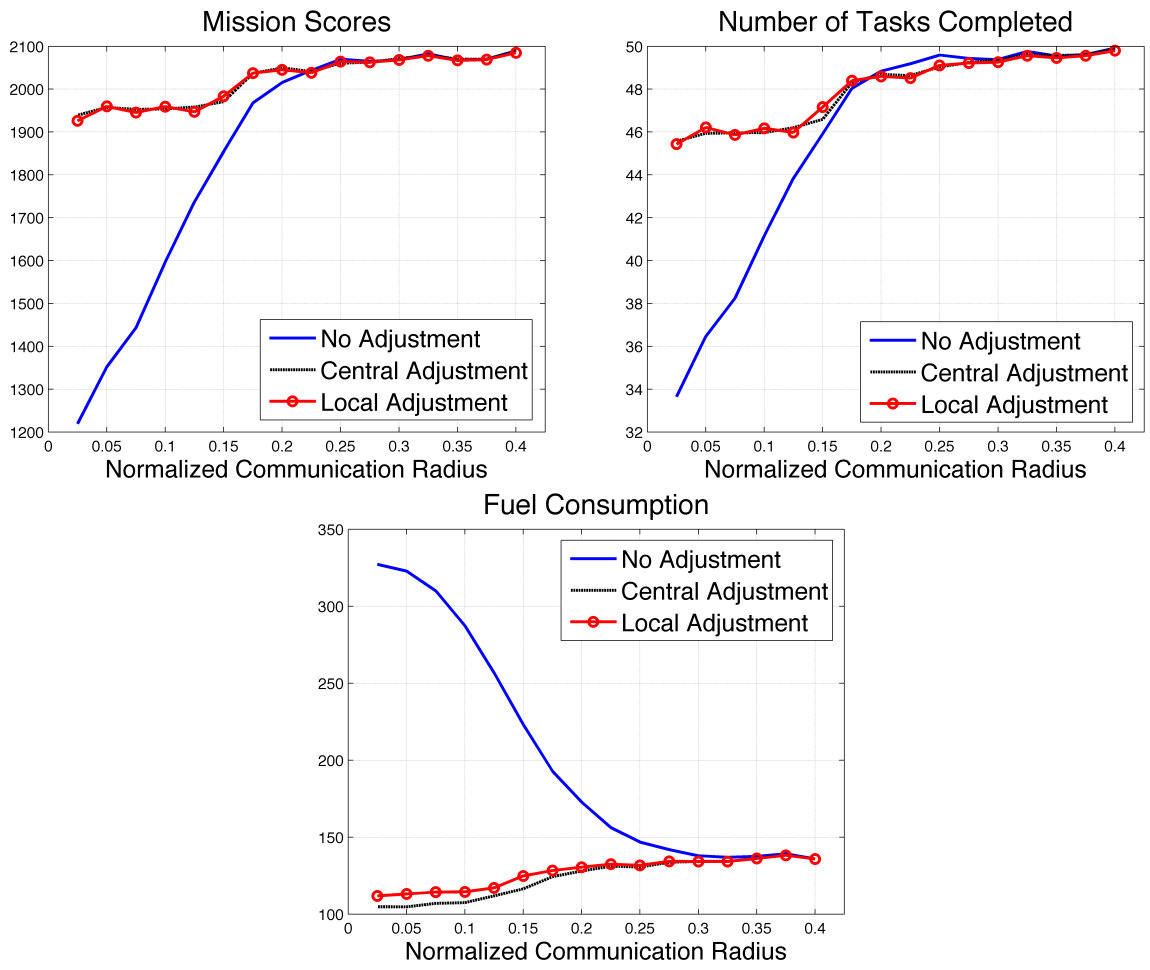


Figure 4-6: Comparison of mission scores, completed tasks and fuel consumption as a function of communication radius for different network handling protocols.

excess fuel usage and the total number of completed tasks is higher since the coordination of the team is improved. The central adjustment method has lower total fuel consumption than the local adjustment method, however, as described in the previous section, this strategy involves redistributing all tasks amongst agents at every replan iteration, and thus the communication requirements associated with this strategy do not scale well with increasing numbers of agents and tasks. The local adjustment method achieves a similar number of completed tasks as the central adjustment method, and although the fuel usage is slightly higher (less efficient paths), the communication overhead required to implement this local adjustment strategy is significantly lower (one message per new task to only one agent).

Experimental Results

Flight experiments for the above scenario were conducted at the MIT Aerospace Controls Lab for a heterogeneous team of 6 agents (3 quadrotor air vehicles and 3 ground vehicles), with a normalized communication radius of 0.1. CBBA with time-varying score functions was used to perform the task allocation and the different replanning architectures with task list adjustments described in the previous sections were implemented. The flight results, shown in Table 4.1, exhibit similar trends to those shown in the simulation results. Both the central and local adjustment methods achieved similar scores and number of tasks completed. The central adjustment method performed slightly better than the local adjustment method, with a lower overall fuel consumption as expected, but with a higher computational and communication overhead. With no task list adjustments the team performance was fairly poor with more fuel consumed and less overall tasks completed.

Table 4.1: Flight Test Results

Adjustment Method	**Score**	**Tasks**	**Fuel**
No Adjustment	897.32	22	111.35
Central Adjustment	1561.44	37	62.79
Local Adjustment	1458.46	34	71.51

Overall, the simulation and experimental flight tests showed that implementing local task list adjustments can drastically improve mission performance in low communication environments, with only marginal increases in required computational overhead, validating the proposed approach. For more details, the reader is referred to [174].

4.4 Ensuring Network Connectivity in Dynamic Environments

The previous section introduced strategies to handle communication disconnects. Some domains, however, require that the team satisfy connectivity requirements during mission execution, since multi-vehicle systems rely on communications to exchange command and control messages and remotely sensed mission data. The inability to communicate sensor data to a base station in real time (e.g. live video) may render the multi-agent system ineffective [104]. Furthermore, failure to properly exchange command and control messages can lead to potentially dangerous system failures. This section presents a cooperative dis-

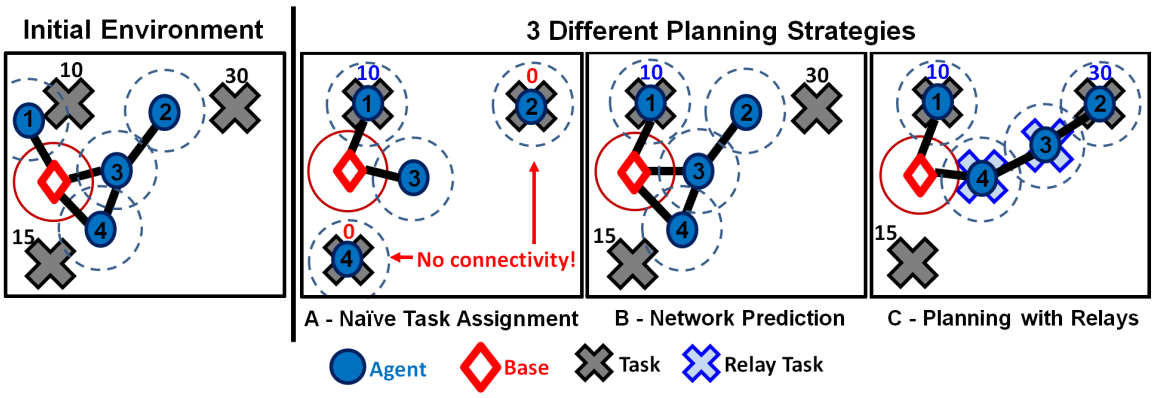


Figure 4-7: Example mission scenario illustrating the benefits of cooperative planning in communication-limited environments. The left box (Initial Environment) shows the initial mission scenario: 4 agents (blue circles) with limited communication range, a base station (red diamond), and 3 tasks of different values (gray x's). The black lines denote connectivity. On the right are 3 different planning strategies. Box A (Naive Task Assignment) shows the team's actions when communication constraints are not considered. Box B (Network Prediction) depicts a conservative solution where each agent detects its own constraint violations and drops tasks, but does not collaborate with other agents explicitly. Finally, Box C (Planning with Relays) shows a cooperative plan where some agents act as communication relays for others, increasing mission performance.

tributed planning algorithm that ensures network connectivity for a team of heterogeneous agents operating in dynamic and communication-limited environments. The algorithm, named CBBA with Relays, builds upon CBBA with time-varying score functions described in 4.2. Information available through the consensus phase of CBBA is leveraged to predict the network topology and to propose relay tasks to repair connectivity violations. Underutilized agents can then be employed as communication relays, improving the range of the team without limiting the scope of the active agents, thus improving mission performance. The CBBA with Relays algorithm ensures network connectivity during task execution but still preserves the distributed and polynomial-time guarantees of CBBA. The next sections describe the scenario and the algorithm in more detail, and results are presented to validate the algorithm through simulations and experimental indoor and outdoor field tests.

4.4.1 Scenario Description

Consider the scenario shown in Fig. 4-7 where agents must perform surveillance around a base station. Vehicles are tasked to select locations to stream live video back to the base, however, agents have a limited communication radius, as in the previous section, and cannot

successfully transmit data if disconnected from the base. The left box of Fig. 4-7 shows the initial environment with 4 agents, a base station, and 3 possible tasks with associated values. The halos around each agent and around the base station depict circles of half of the communication radius, therefore, their intersections signify connectivity. Box A (Naive Task Assignment) shows the results of a plan made without considering communication constraints, where a disconnected network occurs and agents 2 and 4 receive a score of 0 for their tasks since they cannot stream data back to the base.

To prevent disconnects, communication constraints can be explicitly considered in the planning process. Task allocation information, such as task locations and planned execution times, can be leveraged by the agents to predict the network topology at execution. For example, in Box B of Fig. 4-7 (Network Prediction) agents predict the network topology for the proposed assignment and drop tasks that cause disconnects (agents 2 and 4 both drop their assignments and the mission results in only one serviced task). This approach guarantees network connectivity, but is conservative because agents can only accomplish tasks in the local vicinity. An improved solution is to use some agents as communication relays, where data can be transmitted back to the base station through neighboring agents. This requires explicit cooperation between agents to determine where relay tasks are required, which agents should execute these relay tasks, and which agents can execute the main mission tasks. Box C of Fig. 4-7 (Planning with Relays) illustrates this cooperative scenario, where agents predict the network connectivity, detect the potential disconnects, create relay tasks that fix these disconnects, and propose the relay tasks to the rest of the team. Here, Agents 3 and 4 drop their assignments to service relay tasks proposed by Agent 2. This results in a team capability to accomplish higher value tasks, increasing the overall mission score.

While the cooperative planning strategy proposed in the above scenario improves team performance, its formulation is nontrivial. Predicting the topology over time can be computationally intensive as the network is dynamic. Planning algorithms are highly interdependent because the task assignment and network connectivity prediction processes are closely coupled. This complicates network prediction, even for small perturbations in the agent assignments, making it difficult to optimize assignments given connectivity constraints. The challenge of coordinating a multi-agent team to control its communication network has been widely explored in the literature. Studies have investigated motion planning strategies for

teams of agents using methods such as gradient ascent [70], potential fields [224], reactive control [100], and adaptive strategies [153] to steer vehicles to stay within communication range of each other. Other works have explored the problem of optimizing the deployment of designated mobile relay agents to support a network of sensor nodes, using both graph theoretic [101, 161, 225] and network optimization schemes [61].

The CBBA with Relays algorithm presented in this section differs from previous studies by simultaneously optimizing relay and task assignments, instead of preallocating agents to specific roles and then solving decoupled task assignment and network connectivity planning problems. By explicitly coupling the task assignment and relay planning processes, the team is able to better optimize the use of agent resources given the current mission needs, leading to improved performance and added flexibility in real-time dynamic mission scenarios.

4.4.2 CBBA with Relays

The purpose of the CBBA with Relays algorithm is to efficiently allocate agents to tasks while ensuring that the network remains connected to a predefined *base station* during task execution. In this problem, the assumption of agent independence described in Section 2.1.3 and summarized in Eq. (2.4) no longer holds, since the scores agents receive for doing tasks are now dependent on maintaining network connectivity, and thus are functions of other agents' positions over time. However, within the distributed CBBA planning framework, agents have access to other agents' assignments and proposed task times shared through the consensus phase of CBBA, and this information can be leveraged by each agent to predict network connectivity at the agent's proposed task execution times to determine if the proposed tasks will be connected to the base station. As described in Box C of Fig. 4-7, the network connectivity of the team can be adjusted in real-time by using free agents as relays. The CBBA with Relays algorithm achieves this connectivity by leveraging the task allocation capabilities of CBBA with an outer loop that enforces connectivity constraints. In particular, given a set of initial tasks $\mathcal{J}_i$ that each agent i can bid on, and a set of required relay tasks $\mathcal{R}$, CBBA can be used to solve the allocation problem, creating assignments $\mathcal{A}$ of agent-task pairs (here $\mathcal{A}$ is a set of valid assignments extracted from $(\mathbf{x}, \boldsymbol{\tau})$ returned by CBBA). Since CBBA does not guarantee that all available tasks will be assigned, the assignment from a single iteration of CBBA may result in a disconnected network. Network prediction after a CBBA execution can be used to determine if an

assigned task will cause the network to become disconnected at its execution time. Each disconnected task *may* be removed from the task set that the current winning agent i is able to bid on in future iterations. The form of the function that decides if a task will be removed from the available task set is a nontrivial decision that has important implications for both performance and convergence rate (further discussion provided later in this section). For all remaining assigned (non-relay) tasks that are disconnected from the base, new relay tasks are introduced. The criteria for placing these relays is that, if assigned, they would create a connected network. The process then repeats with CBBA generating a new assignment over the new task space that includes all available tasks and newly placed relays. The algorithm converges when CBBA returns a connected assignment with all current relay tasks assigned.

The full CBBA with Relays algorithm is presented in Algorithms 4 and 5. For clarification, a few functions and notation elements will be defined first:

- J_i is the subset of tasks that are currently available to agent i. $\bar{\mathcal{J}} = \{\mathcal{J}_1, \ldots, \mathcal{J}_{N_a}, \mathcal{R}\}$ describes the current available tasks for the team (individual agent available task sets J_i and relay tasks $\mathcal{R}$).

- Winning-Agent(j) returns the current winning agent for task j.

- Dependent-Relays(j) returns the indexes of all relay tasks that are required to be assigned for task j to be connected to the base.

- Dependent-Tasks(r) returns the indexes of all tasks that rely on relay task r being serviced.

- Keep-Task($j, \mathcal{A}$) returns true or false indicating whether j should be dropped. Keep-Task($j, \mathcal{A}$) can be deterministic or stochastic, but must have the property that repeated calling to the function will eventually return false with probability 1 (required for algorithm convergence).

- Place-Relays($j, \bar{\mathcal{J}}, \mathcal{A}$) creates relay tasks required to connect a disconnected task j to the base station. The function is responsible for specifying appropriate locations, values, and time-windows for these relay tasks (further discussion is provided below).

- Predict-Disconnects($\bar{\mathcal{J}}, \mathcal{A}$) returns a set of tasks that will be disconnected at the time of their execution given the proposed assignment $\mathcal{A}$. This function iterates over the

Algorithm 4 CBBA-RELAYS($\mathcal{I}, \mathcal{J}$)

1: $\mathcal{J}_i = \mathcal{J},\ \forall i \in \mathcal{I}$; $\mathcal{R} = \emptyset$
2: $\bar{\mathcal{J}} = \{\mathcal{J}_1, \ldots, \mathcal{J}_{N_a}, \mathcal{R}\}$
3: **while** $\neg$ *converged* **do**
4: $\quad \mathcal{A} \leftarrow$ CBBA($\mathcal{I}, \bar{\mathcal{J}}$) (See Algorithm 1)
5: $\quad$ **for** $(r \in \mathcal{R})$ & $(r \notin \mathcal{A})$ **do**
6: $\quad\quad (\bar{\mathcal{J}}', \mathcal{A}') \leftarrow$ PRUNE-TASK-SPACE(r, $\bar{\mathcal{J}}$, $\mathcal{A}$)
7: $\quad$ **end for**
8: $\quad \mathcal{J}_{disconnected} \leftarrow$ Predict-Disconnects($\bar{\mathcal{J}}'$, $\mathcal{A}'$)
9: $\quad$ **for** $j \in \mathcal{J}_{disconnected}$ **do**
10: $\quad\quad \bar{\mathcal{J}}'' \leftarrow$ Place-Relays(j, $\bar{\mathcal{J}}'$, $\mathcal{A}'$)
11: $\quad$ **end for**
12: $\quad$ **if** $(\bar{\mathcal{J}}'' = \bar{\mathcal{J}})$ & $(\mathcal{A}' = \mathcal{A})$ **then**
13: $\quad\quad$ *converged* $\leftarrow$ true
14: $\quad$ **end if**
15: $\quad \bar{\mathcal{J}} \leftarrow \bar{\mathcal{J}}''$; $\mathcal{A} \leftarrow \mathcal{A}'$
16: **end while**
17: **return** $\mathcal{A}$

assigned tasks and uses the information available in the algorithm to predict the network structure and detect disconnects for each assigned task.

A few important algorithmic considerations are described next. In Algorithm 4, line 4, CBBA with time-varying score functions presented in Section 4.2 (see Algorithm 1) is used as a black box that produces assignments given the set of agents $\mathcal{I}$ and a task list $\bar{\mathcal{J}}$ including the initial tasks $\mathcal{J}$ and relay tasks $\mathcal{R}$. The additional algorithmic pieces of Algorithm 4 enforce agent cooperation and network connectivity constraints, such that, with probability 1, CBBA will return a connected assignment. An important point to note is that, in the Predict-Disconnects algorithm, the network prediction for each assigned task j only involves agents that are *currently executing tasks* at the time of task j (termed "active agents"). This is because "inactive agents" are subject to change their schedules to satisfy relay tasks, and this would invalidate the network prediction if they were included. Another important observation is that this network prediction step can be performed locally by each agent using information available through the CBBA consensus phase, and each agent only needs to check network connectivity during the task execution times of its assigned tasks j, as opposed to most common approaches that involve discretizing time over the entire duration of the mission. By performing these network predictions *locally* and only at *select crucial mission times*, the computation associated with this algorithm remains distributed and tractable.

The Place-Relays algorithm is responsible for creating relay tasks with locations and

Algorithm 5 PRUNE-TASK-SPACE(r, $\bar{\mathcal{J}}$, $\mathcal{A}$)

$\bar{\mathcal{J}} \leftarrow \bar{\mathcal{J}} \setminus \{r\}$
for $j \in$ Dependent-Tasks(r) **do**
  for $r' \in$ Dependent-Relays(j) **do**
    Dependent-Tasks(r') $\leftarrow$ Dependent-Tasks(r') $\setminus \{j\}$
    if Dependent-Tasks(r') $= \emptyset$ **then**
      $\bar{\mathcal{J}} \leftarrow \bar{\mathcal{J}} \setminus \{r'\}$
      $\mathcal{A} \leftarrow \mathcal{A} \setminus \{r'\}$ **if** $r' \in \mathcal{A}$
    end if
  end for
  Dependent-Relays(j) $\leftarrow \emptyset$
  $keep \leftarrow$ Keep-Task($j, \mathcal{A}$)
  if $\neg keep$ **then**
    $\mathcal{A} \leftarrow \mathcal{A} \setminus \{j\}$
    $\mathcal{J}_{Winning-Agent(j)} \leftarrow \mathcal{J}_{Winning-Agent(j)} \setminus \{j\}$
  end if
end for
return $(\bar{\mathcal{J}}', \mathcal{A}')$

time-windows that ensure connectivity for the main task they are designed to connect. This process can be executed locally by each agent assigned to a disconnected task, and is dependent on the environment and connectivity models available for that agent. For the results in this thesis, the connectivity was modeled as a function of communication radius, and the relays were placed between the closest pair of agents that would ensure connectivity for the disconnected task j given the network prediction. The time-windows and durations were set such that the relays would be in place during task j's proposed execution time. The reward value for the relays was equal to the bid made on task j divided by the number of relays placed to create the connection. Note that during mission execution, agents performing relay tasks do not actually receive a score, but they do incur fuel penalties. In a sense, the relay scores are "virtual scores", but setting them to fractions of task j's bid guarantees that the total mission scores obtained (scores for connected tasks minus all fuel usage) will never be negative. Current research is exploring alternate ways to place relays that explicitly consider link capacities and bit error rates, rather than simply shortest relay distance [116, 117].

As alluded to above, the form of the Keep-Task function impacts the convergence performance of the algorithm. In order to guarantee convergence of CBBA with Relays, agents can only be allowed to propose relays for a disconnected task for a finite number of iterations. Eventually, the agent will have to "give up" and add the task to its "do-not-allow" list. In this way, the number of task choices available to the agents decreases as the algorithm

iterations proceed. As a reminder, relay tasks are associated with the actual tasks that they are designed to connect, and are only valid for one iteration of the CBBA with Relays algorithm (any unassigned relay tasks are deleted), so as the number of actual task choices decreases, so will the number of relay tasks. Given this diminishing task space, the CBBA with Relays algorithm will converge, since eventually every agent will have no remaining valid tasks to bid on. Therefore, to guarantee convergence of CBBA with Relays, the only requirement is that the Keep-Task function must return false with positive probability for every agent-task pair, so that successive repeated calls to the function will eventually return false with probability 1 (and thus the agent will not be able to bid on that task again).

A trivial form of the Keep-Task function is to always return false (i.e. agents only get to try for a disconnected task for 1 iteration), since this strategy has a positive probability of returning false which is required for convergence. This has the effect of making all agents drop their disconnected tasks simultaneously, which causes coordination problems amongst the agents. As an illustrative example, consider a stale-mate case, where 2 agents have tasks assigned that are both disconnected from the base and thus both agents require relays. Since both agents are occupied, they cannot satisfy the relay requirement for the other agent. As a result, in the next iteration, both agents drop their tasks (and are never allowed to bid on them again), and select other assignments instead. A more coordinated approach would have resulted if one agent had dropped its assignment and satisfied the relay requirement of the other agent. However, in this situation, it is difficult to predict which agent should drop its task first and satisfy the relay requirement for the other agent. An obvious method would be for each agent to solve a centralized problem involving both agents, and to select the action that leads to the highest team score. This method does not scale well, however, because this type of stalemate can happen with many more than 2 agents simultaneously and with more than a single task per agent. Furthermore, not only do the distributed agents need to predict their own plans, they would also need to model the decisions of all other agents (whether they will drop their tasks, and what assignments they will select after a possible drop), making the problem intractable even with small teams of agents.

As an alternative, a stochastic approach can be used where agents probabilistically keep tasks for subsequent iterations. In this work, a probabilistic rule is employed, where the function returns false with probability p that is proportional to the bid value on task j (normalized against the maximum obtainable task score). This stochastic aspect of the

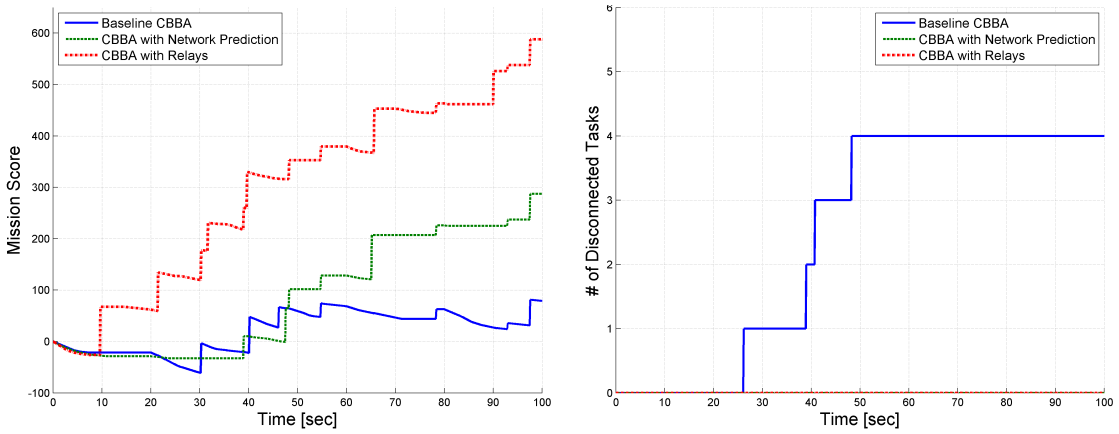
algorithm prevents all agents from releasing their tasks simultaneously, thus breaking the symmetry associated with conflicted assignments, and allowing other agents the opportunity to bid on relay tasks. The probability of dropping a disconnected task is proportional to the current bid of that task, so, on average, higher value tasks are kept around longer. For the environments considered in this thesis, this heuristic approach works well, has low computational overhead, and is guaranteed to converge since there is a positive probability p that the Keep-Task function will return false at any given iteration, and thus that the disconnected tasks will be dropped (forever) from agents' assignments. Therefore all disconnected tasks will eventually be unavailable to all agents with probability 1. It is worth noting, however, that for smaller or highly structured domains more complex rules could be developed that improve algorithm performance, but for general scenarios of interest this probabilistic strategy performs well and, in practice, the CBBA with Relays algorithm converged rapidly, enabling real-time applicability.

4.4.3 Example Applications

This section presents simulation and hardware results that validate the performance of CBBA with Relays. The algorithm is compared against the two other planning methods discussed above (see Figure 4-7): the first is baseline CBBA which does not include communication constraints in the planning process, where the algorithm fails to detect potential disconnects leading to reduced mission performance (Fig. 4-7(a)); and the second algorithm, termed CBBA with Network Prediction, involves conservatively dropping tasks that are likely to cause network disconnects (Fig. 4-7(b)). In this algorithm, agents execute CBBA to obtain task assignments and then predict the network structure to check whether connectivity constraints will be violated. Tasks that will cause disconnects are dropped and the corresponding agents are not allowed to rebid on them. The algorithm then executes CBBA again, iterating until the team assignment remains constant and no more tasks are being dropped.

Simulation Results

This section shows simulation results for a 6 agent dynamic mission, where the simulation architecture consisted of a mission control center, a network detector, the distributed planning algorithms, and local agent simulations as described before (see Figure 4-3). In this



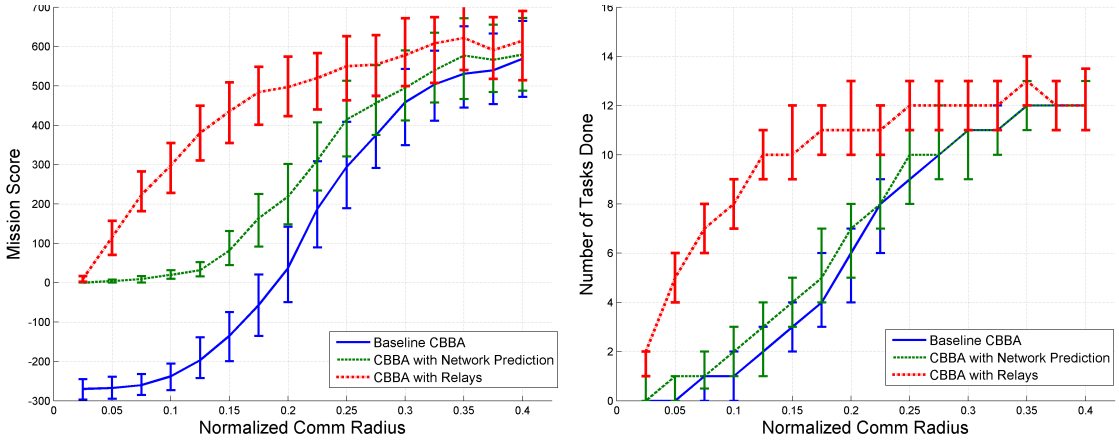
(a) Mission Scores

(b) Number of Disconnected Tasks

Figure 4-8: Results for a single simulation run of a 6 agent mission, comparing Baseline CBBA, CBBA with Network Prediction, and CBBA with Relays. The plots show the achieved mission scores as a function of time, and the number of disconnected tasks throughout the mission execution.

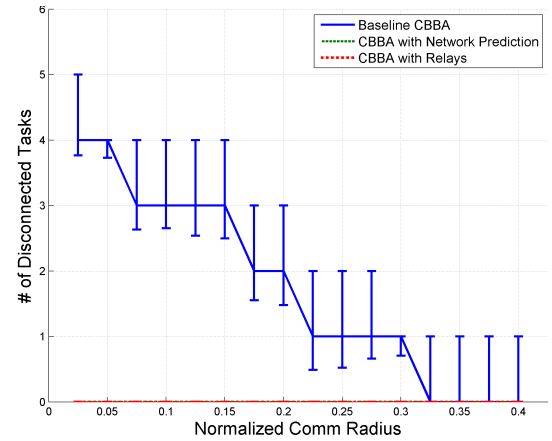
scenario, tasks appeared at a constant rate, with randomly distributed positions, and values proportional to the distance from the base station (representing the fact that information further from the base is more valuable since it is less likely to be known). The communication radius for the agents was set to 20% of the environment, and agents were only able to execute tasks and receive rewards for them if they were connected to the base station at the time of task execution. If they were not connected to the base, then they had to abandon the corresponding task and move to the next one in their task list. Figure 4-8(a) shows the mission scores as a function of time and Figure 4-8(b) shows the number of disconnected tasks during execution. As seen in the plots, Baseline CBBA causes significant network disconnects leading to poor performance. CBBA with Network Prediction improves the mission performance by preventing network disconnects, but is very conservative in the tasks it schedules achieving only marginally higher performance. CBBA with Relays clearly outperforms the other algorithms by allowing cooperative task execution, achieving a higher score throughout the mission and ensuring connectivity during task execution.

To further analyze the performance of CBBA with Relays, a Monte Carlo simulation was implemented, using the scenario described above but varying the communication radius for the agents. Figure 4-9 shows the mean mission scores, number of tasks done, number of disconnected tasks, and planner run-time as a function of communication radius, with 25%

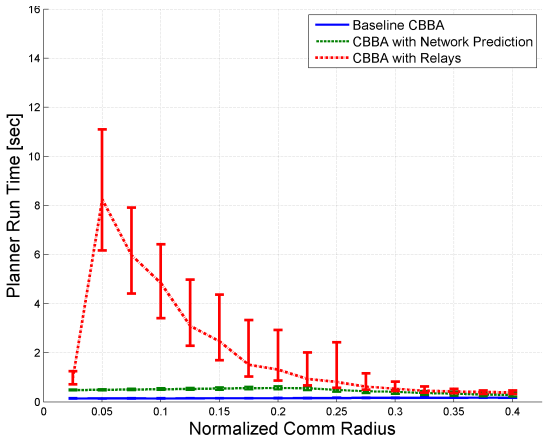


(a) Mission Scores

(b) Number of Tasks Done



(c) Average Disconnected Tasks

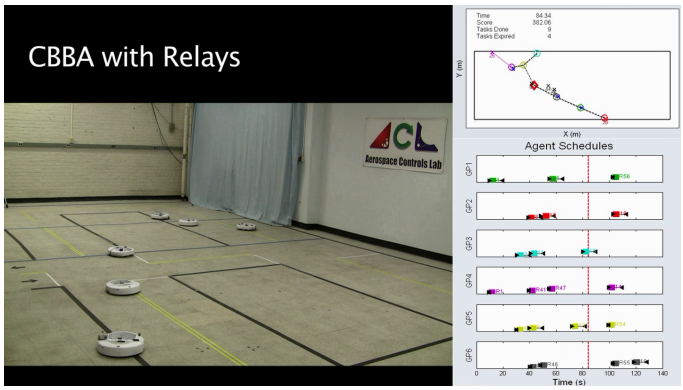


(d) Average Planner Run Time

Figure 4-9: Monte Carlo simulation results for a 6 agent mission, comparing the performance of Baseline CBBA, CBBA with Network Prediction, and CBBA with Relays. The plots show the mission scores, the number of tasks done, the average number of disconnected tasks, and the average planner run time, as a function of normalized communication radius.

and 75% error bars shown on the plots. Once again CBBA with Relays achieves higher mission performance than the other two approaches, with higher scores (Fig. 4-9(a)) and greater number of tasks done (Fig. 4-9(b)). Both CBBA with Network Prediction and Baseline CBBA achieve similar number of tasks performed, however, in the baseline case agents attempt to execute tasks which will cause disconnects, and thus waste fuel without being able to perform the far tasks, thus leading to lower mission scores. Both CBBA with Relays and CBBA with Network Prediction ensure network connectivity during task execution (Fig. 4-9(c)). The planner run-time for CBBA with Relays is higher than that for the other two algorithms, and is highest when several relays must be assigned (Fig. 4-9(d)).

(a) iRobot Create



(b) Indoor Experimental Mission

Figure 4-10: Real-time indoor autonomous vehicle experiments, at the MIT Aerospace Controls Lab, demonstrating CBBA with Relays. The figures show a robotic platform consisting of the iRobot Create and an indoor experimental 6 agent mission using the CBBA with Relays algorithm.

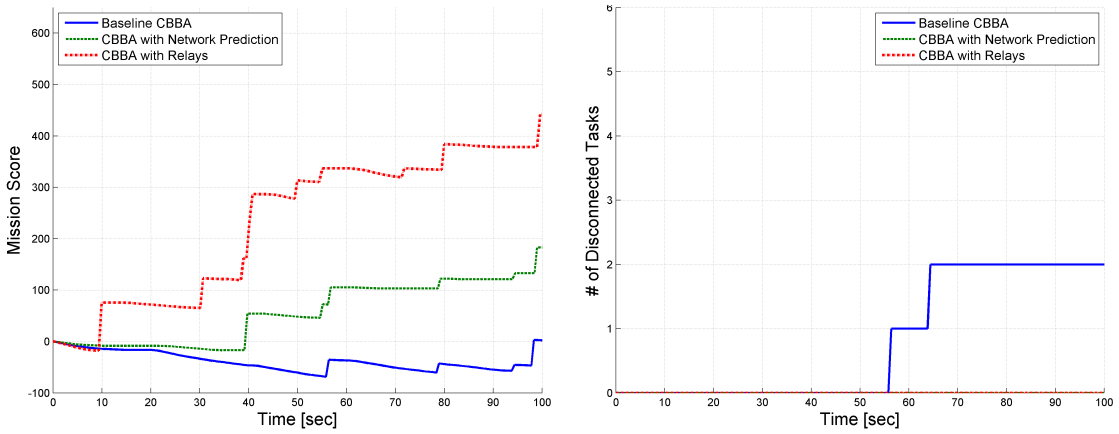
It drops off as the connectivity improves and also at really low communication radii[5]. For the 6 agent missions considered here, at a normalized communication radii higher than 0.3 (i.e. 30% of the theater of operation) the network remains mostly connected and all the algorithms achieve similar performance.

Experimental Results

To validate the real-time performance of CBBA with Relays, hardware experiments were conducted at the MIT Aerospace Control Lab. For these missions, the robotic platform consisted of iRobot Creates (Fig. 4-10(a)), equipped with an xBee-PRO wireless module for communication, where high-bandwidth position and attitude data was provided by a motion-capture system. Figure 4-10(b) shows a snapshot of a 6 agent mission, where the front agent is connected to the base (further back) through two relay agents. The hardware results for this mission scenario in Figure 4-11 are similar to those discussed in Figure 4-8, demonstrating the real-time applicability of the algorithm. Once again the mission scores are highest using CBBA with Relays, and the algorithm ensures that no tasks are disconnected throughout the mission.

To demonstrate the algorithm in a more operationally realistic setting, further experiments were conducted in an outdoor flight testing environment using a team of UAVs,

[5]At very low communication radii the number of relays required is greater than the available agents, making the tasks infeasible immediately, thus lowering the plan time.

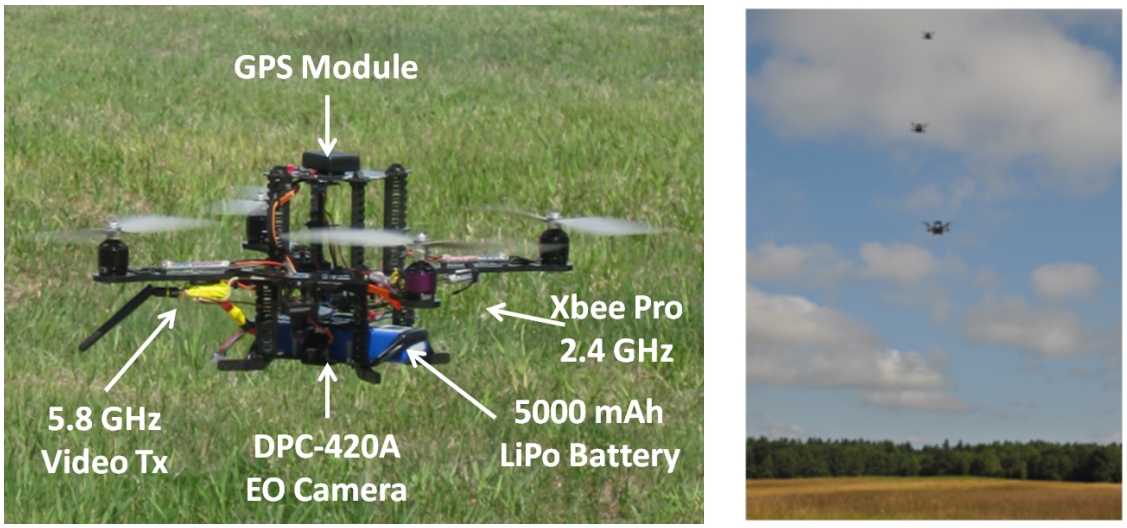


(a) Mission Scores

(b) Number of Disconnected Tasks

Figure 4-11: Real-time indoor experimental results for a 6 agent mission, comparing Baseline CBBA, CBBA with Network Prediction, and CBBA with Relays. The plots show the achieved mission scores as a function of time, and the number of disconnected tasks throughout the mission execution.

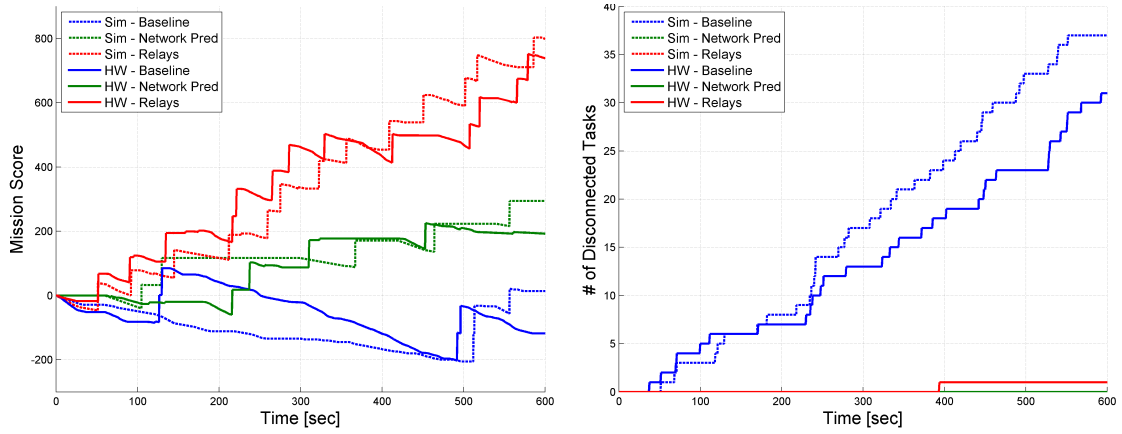
consisting of Ascending Technologies Pelican Quadrotors (Fig. 4-12(a)) [118]. Each vehicle was 2.5 lbs, had a flight endurance of 18 minutes, and was capable of GPS waypoint navigation while communicating with the base station using a Digi-Mesh XBee 2.4 GHz radio module. The missions were performed in software simulation and then executed in outdoor flight tests in restricted airspace at a military facility (see Fig. 4-12(b)). Figure 4-13 shows the results of the flight experiments (solid lines) along with their simulation predictions (dotted lines). As before, using CBBA with Relays agents collaborated to accomplish valuable tasks in the search area, achieving scores which were more than double of those obtained using the non-cooperative CBBA with Network Prediction strategy, and significantly higher than the Baseline CBBA algorithm which achieved negative scores (since most tasks resulted in failures). One disconnect did occur using CBBA with Relays in flight testing. Due to modeling inaccuracies, imperfect state estimation, and environmental effects, one of the UAVs actually reached a task ahead of the predicted time and therefore started it early. Once finished, it moved on early to the next task, allowing another task dependent on it at the predicted time to disconnect. A modification to the algorithm has since been proposed which forces agents to wait until the predicted time to start their tasks in order to maintain the planning schedule. This event highlights the importance of hardware testing in environments less controlled than simulation. The results also show that the algorithm,



(a) Ascending Technologies Pelican Quadrotor

(b) Outdoor Flight Experiment

Figure 4-12: Real-time autonomous vehicle outdoor flight experiments demonstrating CBBA with Relays. The figures show a quadrotor UAV platform consisting of the Ascending Technologies Pelican Quadrotor, and an outdoor experimental 3 agent mission using the CBBA with Relays algorithm.



(a) Mission Scores

(b) Number of Disconnected Tasks

Figure 4-13: Real-time outdoor flight test results for a 3 agent mission, comparing Baseline CBBA, CBBA with Network Prediction, and CBBA with Relays. The plots show the achieved mission scores as a function of time, and the number of disconnected tasks throughout the mission execution.

even without the fix, generally worked well in conditions with moderate uncertainty, and closely followed the trends observed in simulation. More details and results for the CBBA with Relays algorithm are available in [118, 170, 173].

4.5 Summary

This chapter presented several key extensions to the baseline CBBA algorithm proposed in [58] to address dynamic mission planning in realistic environments. Section 4.2 proposed an extension to CBBA that enabled optimization given time-varying score functions and dynamic mission scenarios (CBBA with Time-Varying Score Functions). Section 4.3 presented strategies to ensure conflict-free solutions in the presence of network disconnects through local task space partitioning methods. And, Section 4.4 proposed a cooperative planning algorithm (CBBA with Relays), that builds upon the baseline CBBA framework to enable cooperative mission execution in communication-limited environments through the use of relay tasks, thus improving mission performance.

The algorithms presented in this chapter focused mostly on deterministic planning scenarios. As mentioned in Section 2.1.4, a major consideration is that planning algorithms rely on underlying system models, which are often subject to uncertainty, and discrepancies between these planner models and the actual system dynamics can cause significant degradations in mission performance. The next chapters address this issue by proposing stochastic planning extensions to the distributed CBBA algorithm that enable agents to hedge against parameter uncertainty using the different stochastic metrics discussed in Section 2.1.4.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Distributed Planning in Uncertain Domains

As described in Chapter 2, planning algorithms rely on underlying system models which are often subject to uncertainty. This uncertainty can result from many sources including: inaccurate modeling due to simplifications, assumptions, and/or parameter errors; fundamentally nondeterministic processes (e.g. sensor readings, stochastic dynamics); and dynamic local information changes. As discrepancies between the planner models and the actual system dynamics increase, mission performance typically degrades. The impact of these discrepancies on the overall quality of the plan is usually hard to quantify in advance due to nonlinear effects, coupling between tasks and agents, and interdependencies between system constraints (e.g. longer-than-expected task durations can impact the arrival times of subsequent tasks in the path, as previously described in Figure 2-3). However, if uncertainty models of planning parameters are available they can be leveraged within the planning framework to create *robust plans* that explicitly hedge against the inherent uncertainty to improve mission performance. This chapter proposes an extension to the deterministic CBBA with time-varying score functions algorithm described in Section 4.2, which enables the use of different stochastic metrics to mitigate the effects of parameter uncertainty given probabilistic agent and task models. The Robust CBBA algorithm is described in detail in the following sections, and results are presented to validate this robust distributed real-time framework, showing improved performance for teams of networked agents operating in uncertain and dynamic environments.

5.1 Stochastic Distributed Problem Formulation

Consider the problem definition introduced in Eq. (2.6) but where the planning parameters $\boldsymbol{\theta}$ are random variables ($\boldsymbol{\theta}$ is a vector of planning parameters including elements such as agent velocities, task durations, wind speed and direction, etc.). Assume that models of the uncertainty are available, where $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ with distribution given by the joint probability density function (PDF), $\mathbf{f}(\boldsymbol{\theta})$. The goal of stochastic planning is to use the information provided in $\mathbf{f}(\boldsymbol{\theta})$ to create plans that explicitly account for the *variability* and *coupling* of the uncertain parameters in the score functions $\mathbf{c}_{ij}$.

5.1.1 Uncertain Parameter Types

The uncertain planning parameters $\boldsymbol{\theta}$ involve 3 main types of variables: agent specific parameters, task specific parameters, and environment parameters. Agent specific parameters include variables such as fuel consumption and cruise velocities, which are specific to each agent and therefore do not usually affect the performance of other agents. Task specific parameters could include stochastic task durations, uncertain task locations (e.g. tracking a moving target), and other random variables that can usually be associated with individual tasks. Environment parameters include variables that affect all agents performing tasks in a certain area (for example, wind velocity and direction), or could be localized to specific areas (e.g. wind in a canyon may be different than wind over a plain). The uncertain parameters could include combinations of these groups as well. For example, an agent executing a certain task may have a different stochastic distribution over task duration than a different agent performing the same task. There could also be coupling between different parameters. For example, long task durations for some tasks may be correlated with long task durations for other tasks (e.g. a bad sensor reading could impact target identification time as well as the time required to search for and localize the target).

In general, accounting for all these different coupling effects is difficult, particularly in distributed planning environments, since it would require that agents consider the coupling between parameters with other agents when making their individual plans. An assumption that can be made to reduce the complexity of the planning problem is that the uncertain parameters can be partitioned among the agents so that agent scores are independent. This assumption is valid for agent specific parameters (e.g. distributions over agent velocities

affect agents individually), and is also true for task specific parameters assuming conflict-free solutions since no two agents are assigned to the same task[1]. Using this assumption, agents can plan independently using their own situational awareness and planning parameter distributions, instead of having to coordinate with other agents during the planning process to account for coupling between agent scores. Cases that produce coupling between agent score functions such as environment parameters (e.g. wind direction) are much more difficult to consider when planning in distributed environments, requiring consensus protocol modifications and additional communication requirements between agents to account for coupling between agent assignments during the task selection process, and are therefore beyond the scope of this thesis.

5.1.2 General Stochastic Distributed Framework

To enable real-time robust distributed planning for networked teams operating in stochastic environments, the main questions to consider are: how does the uncertainty in planning parameters propagate through each agent's plan due to coupling between the tasks (e.g. temporal coupling), how can the stochastic optimization be distributed given the additional complexity associated with uncertain planning parameters, and how can computational tractability be maintained to ensure real-time performance given the additional planning dimensions associated with the uncertain planning parameters? The Robust CBBA algorithm described throughout this chapter addresses these three key concerns, enabling real-time robust planning within a distributed framework. To address the first question, the Robust CBBA algorithm employs different stochastic metrics to propagate the effects of parameter uncertainty through the mission score. Equation (5.1) shows the general problem formulation, where the stochastic metric $\mathbb{M}_{\boldsymbol{\theta}}(\cdot)$ is used to quantify the effect of uncertainty on the mission score. There are several stochastic metrics that have been considered throughout the literature [25, 28, 33], however, most of these have been considered within a centralized

[1]During the CBBA planning process, two agents can have the same task in their bundles, and therefore their stochastic scores would not be independent. This coupling, however, is similar to the deterministic case where agents' bundle scores could be inaccurate mid-iteration, since one of the agents would eventually have to drop the task. Therefore, as long as the algorithm returns a conflict-free solution, the final result will be such that agent score distributions are independent.

optimization framework (as described in Eq. (5.1)).

$$\begin{aligned}
\max_{\mathbf{x},\boldsymbol{\tau}} \quad & \mathbb{M}_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta}) \; x_{ij} \right) \right\} \\
\text{s.t.} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_i, \quad \forall i \in \mathcal{I} \\
& \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\
& x_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \\
& \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J}
\end{aligned} \tag{5.1}$$

In order to use these stochastic metrics within the Robust CBBA distributed planning framework, the sum over agent scores must be extracted from the metric $\mathbb{M}_{\boldsymbol{\theta}}(\cdot)$ in Eq. (5.1) (i.e. no coupling between agents due to the stochastic metric). If this is possible, then the centralized optimization in Eq. (5.1) can be broken down into N_a distributable sub-problems of the form shown in Eq. (5.2), where each agent i can solve its own individual optimization.

$$\begin{aligned}
\max_{\mathbf{x}_i,\boldsymbol{\tau}_i} \quad & \mathbb{M}_{\boldsymbol{\theta}} \left\{ \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta}) \; x_{ij} \right\} \\
\text{s.t.} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_i, \quad \forall i \in \mathcal{I} \\
& \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\
& x_{ij} \in \{0,1\}, \quad \forall j \in \mathcal{J} \\
& \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall j \in \mathcal{J}
\end{aligned} \tag{5.2}$$

Given the distributed stochastic problem statement of Eq. (5.2), the only coupling between agents is through the conflict-free constraint, as specified previously in Eq. (2.3), which can be handled through the CBBA consensus protocol as before. Since agent scores are assumed to be independent, if the stochastic metric is distributable, agents can build their bundles independently using their own situational awareness and planning parameter distributions, instead of having to communicate with other agents while building their bundles to account for coupling between agent scores. As a result, the number of messages required between

agents to come to consensus on plans is only associated with ensuring that plans remain conflict free, and is thus equivalent to the deterministic CBBA message requirements.

5.1.3 Distributing Stochastic Metrics

In this section, we revisit the stochastic metrics introduced in Section 2.1.4 and show how these can be used within the distributed Robust CBBA planning framework. The first metric considered is the expected-value metric which optimizes average mission performance given the stochasticity in the system [28]. Using this metric, a stochastic version of Eq. (2.6) can be written as follows,

$$\begin{aligned}
\max_{\mathbf{x},\boldsymbol{\tau}} \quad & \mathbb{E}_{\boldsymbol{\theta}}\left\{\sum_{i=1}^{N_a}\left(\sum_{j=1}^{N_t}\mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)),\boldsymbol{\theta})\ x_{ij}\right)\right\} \\
\text{s.t.} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_i, \quad \forall i \in \mathcal{I} \\
& \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\
& x_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{I}\times\mathcal{J} \\
& \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall (i,j) \in \mathcal{I}\times\mathcal{J}
\end{aligned} \tag{5.3}$$

Note that optimizing Eq. (5.3) is not the same as deterministically planning using the mean values of uncertain planning parameters $\bar{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{\theta}}\{\boldsymbol{\theta}\}$. Using just the mean values fails to capture the coupling of uncertainty in the score function, leading to potentially poor planning performance. Leveraging the fact that the expected value of a sum of random variables is equivalent to the sum of the expected values, i.e.

$$\mathbb{E}_{\boldsymbol{\theta}}\left\{\sum_{i=1}^{N_a}\left(\sum_{j=1}^{N_t}\mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)),\boldsymbol{\theta})\ x_{ij}\right)\right\} = \sum_{i=1}^{N_a}\ \mathbb{E}_{\boldsymbol{\theta}}\left\{\sum_{j=1}^{N_t}\mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)),\boldsymbol{\theta})\ x_{ij}\right\} \tag{5.4}$$

the sum over agents can be moved outside of the stochastic metric (note that Eq. (5.4) is true even when agent score functions are not independent). Thus the centralized problem can be decomposed into sub-problems of the form,

$$\max_{\mathbf{x}_i,\boldsymbol{\tau}_i} \quad \mathbb{E}_{\boldsymbol{\theta}}\left\{\sum_{j=1}^{N_t}\mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)),\boldsymbol{\theta})\ x_{ij}\right\} \tag{5.5}$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_i, \\
& \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\
& x_{ij} \in \{0,1\}, \quad \forall j \in \mathcal{J} \\
& \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall j \in \mathcal{J}
\end{aligned}
$$

where each agent i must solve the optimization in Eq. (5.5), and summing over the local agent expected-value scores gives the global expected-value mission score.

The second metric considered in Section 2.1.4 involves maximizing the worst case performance of the system, which can be used when the current mission tolerance to failure is very low, requiring stronger performance guarantees than those provided by expected value planning. Using this metric, Eq. (2.6) becomes,

$$
\begin{aligned}
\max_{\mathbf{x},\boldsymbol{\tau}} \quad & \min_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij} \right) \right\} \qquad (5.6) \\
\text{s.t.} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_i, \quad \forall i \in \mathcal{I} \\
& \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\
& x_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \\
& \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J}
\end{aligned}
$$

Optimizing Eq. (5.6) guarantees that the plan execution will result in a score *no worse* than that predicted by the algorithm. Therefore the score returned by the planner is a lower bound on the attainable mission performance. Unfortunately, extracting the sum over agents out of the stochastic metric is not as straightforward as in the expected-value case. This is because, given a sum of non-independent random variables, the value of the uncertain parameters that cause each random variable to take on their minimum value could be different. Therefore, it may not be possible to find a single realization of the uncertainty $\boldsymbol{\theta}$ that causes all the random variables to take on their lowest values simultaneously. Thus, minimizing over each random variable separately and then summing over these minimum values could give a lower score than minimizing over the sum of the random variables. In

particular, extracting the sum over agents out of the stochastic metric gives the following relationship,

$$\min_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij} \right) \right\} \geq \sum_{i=1}^{N_a} \min_{\boldsymbol{\theta}} \left\{ \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij} \right\} \quad (5.7)$$

For cases where the agent scores are *independent* random variables (i.e. $\boldsymbol{\theta}$ can be partitioned into disjoint sets between the agent score functions), the inequality in Eq. (5.7) becomes an equality, but in the general case, the right-hand side of Eq. (5.7) provides a lower bound on the minimum attainable mission score given the uncertainty in $\boldsymbol{\theta}$. Breaking the centralized optimization of Eq. (5.6) into sub-problems gives,

$$\begin{aligned} \max_{\mathbf{x}_i, \boldsymbol{\tau}_i} \quad & \min_{\boldsymbol{\theta}} \left\{ \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij} \right\} \\ \text{s.t.} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_i, \\ & \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\ & x_{ij} \in \{0, 1\}, \quad \forall j \in \mathcal{J} \\ & \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall j \in \mathcal{J} \end{aligned} \quad (5.8)$$

where each agent i can solve the optimization in Eq. (5.8) individually, and summing over the local agent scores gives a lower bound on the the worst-case global mission score. The true worst-case mission score may be higher that the score predicted by the distributed planner if there is coupling between the uncertain parameters, but for the cases considered in this thesis, where agent scores are independent, the centralized and distributed problem statements are equivalent, and therefore the distributed planner scores accurately predict the worst-case mission score.

As mentioned in Section 2.1.4, maximizing the worst-case mission score is usually too conservative, and a chance-constrained metric can be used instead, which guarantees that the global mission performance will be at least as good as the proposed plan value within a certain allowable risk threshold. Using the chance-constrained metric, Eq. (2.6) can be

modified as follows,

$$
\begin{aligned}
\max_{\mathbf{x},\boldsymbol{\tau}} \quad & y \\
\text{s.t.} \quad & \mathbb{P}_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta}) \; x_{ij} \right) > y \right\} \geq 1 - \epsilon \\
& \sum_{j=1}^{N_t} x_{ij} \leq L_i, \quad \forall i \in \mathcal{I} \\
& \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\
& x_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \\
& \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J}
\end{aligned}
\tag{5.9}
$$

In Eqs. (5.3) and (5.6), which use the expected value and worst-case value metrics respectively, the uncertain variables $\boldsymbol{\theta}$ do not affect any joint constraints between agents. Therefore, extending CBBA to include these stochastic metrics only involves modifying the bundle construction process, and the consensus protocol of CBBA (task consensus phase described in Section 4.1.2), which explicitly handles the joint conflict-free constraint (Eq. (2.3)), can be used without modification as long as *valid bids* are available (recall that, in distributed planning algorithms, consensus protocols are usually required to ensure satisfaction of joint constraints between agents). Unfortunately, the chance-constrained problem formulation considered in Eq. (5.9) includes a joint probability constraint that couples the agents' task assignments (2^{nd} line of Eq. (5.9)). In this formulation, the sum over agents cannot be easily extracted from the stochastic metric, and therefore the decomposition of Eq. (5.2) cannot be employed. As a result, solving this optimization in a distributed manner using the CBBA framework would require modifying the consensus protocol to ensure that the joint probabilistic constraint is met, which is a nontrivial endeavor.

The Robust CBBA algorithm proposed in this chapter specifically optimizes distributable stochastic metrics, such as the expected-value and the worst-case metrics, within a distributed framework. The more complex chance-constrained metric considered in Eq. (5.9) is revisited in Chapter 6, and a robust extension to CBBA is proposed that allows an approximation of the chance-constrained problem to be solved in a distributed fashion. The next section describes how CBBA can be modified to account for uncertainty in planning

parameters, focusing in particular on how stochastic scores can be computed during the bundle building process and how valid bids can be constructed to ensure algorithm convergence.

5.2 Robust Extension to CBBA

5.2.1 Computing Stochastic Scores

Within the distributed CBBA framework, agents select assignments that optimize their own local score functions and then perform consensus amongst each other to resolve conflicts. These score calculations are performed using each agent's local understanding of the planning parameters, and CBBA is guaranteed to converge even when agents have varying situational awareness. In the robust extension to the CBBA algorithm proposed in this section, this property of CBBA is leveraged. In particular, within Robust CBBA, agents use their own local situational awareness of the uncertain planning parameters and associated distributions when building their bundles. As described in the previous section, this requires that agents have knowledge about parameters that affect them only (e.g. agent specific parameters, task parameters for relevant tasks, and environment parameters for relevant operating areas), and irrelevant information that doesn't affect the particular agent scores can safely be ignored. It is important to note that, in situations where agent scores are coupled (for example, through coupled tasks with other agents, or through environment parameters affecting all agents), the distributed CBBA framework can still be used and is still guaranteed to converge, although the performance of the team may decrease if the coupling is not explicitly considered.

When optimizing agent scores, one main issue with evaluating the stochastic planning metrics described in the previous section is that the agent score functions consist of sums of non-independent heterogeneous task scores. Therefore, computing the distribution of an agent's score involves combining task score distributions for all tasks in the agent's assignment in nontrivial ways (e.g. convolution if independent), which is only tractable given very specific distribution types (e.g. i.i.d random variable, Gaussian distributions, exponential-Erlang, etc.). As a result, analytically computing the integrals, convolutions, and coupling effects associated with the stochastic metrics in closed form is usually impossible unless several limiting assumptions on the allowable distributions, uncertainty models, and score

functions are made. Another issue particular to this problem, which complicates these computations even further, is that evaluating scores for agents' paths implicitly involves selecting the optimal task execution times for the current agent's path. Recall that, in the deterministic case, the score that agent i obtains for a given path $\mathbf{p}_i$ is,

$$J_{\mathbf{p}_i} = \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^{\star}(\mathbf{p}_i), \boldsymbol{\theta})\ x_{ij} \tag{5.10}$$

where $x_{ij} = 1$ for all tasks in the path, and where the optimal task execution times $\boldsymbol{\tau}_i^{\star}$ are found by solving,

$$\boldsymbol{\tau}_i^{\star} = \underset{\boldsymbol{\tau}_i}{\operatorname{argmax}}\ \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i), \boldsymbol{\theta})\ x_{ij} \tag{5.11}$$

In stochastic environments, the task execution times are usually random variables which are subject to the uncertainty in the system. This makes the step of computing the "optimal" execution times nontrivial, since these times may be different for different realizations of the uncertainty. Therefore, when optimizing stochastic path scores, the computation must take into account that different optimal execution times may result for a given path subject to the uncertainty in the environment. For example, using the expected-value metric, the stochastic path score is given by,

$$J_{\mathbf{p}_i} = \mathbb{E}_{\boldsymbol{\theta}}\left\{\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^{\star}(\mathbf{p}_i), \boldsymbol{\theta})\ x_{ij}\right\} = \int_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^{\star}(\mathbf{p}_i), \boldsymbol{\theta})\ x_{ij}\right) \mathbf{f}(\boldsymbol{\theta})\ d\boldsymbol{\theta} \tag{5.12}$$

where for each realization of the uncertainty $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ the optimal task execution times $\tau_{ij}^{\star}$ must be determined. Analytically computing these effects is very difficult, motivating the use of sampling methods to approximate these stochastic score calculations. An example of the sampling process used within the Robust CBBA framework is provided in Algorithm 6. Here, numerical techniques that efficiently sample from $\mathbf{f}(\boldsymbol{\theta})$ can be used to approximates the distribution of $\boldsymbol{\theta}$, generating a set of representative samples, $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$ (Alg. 6, line 1), with corresponding probability weights $\{w_1, \ldots, w_N\}$ (Alg. 6, line 2), where $\sum_{k=1}^{N} w_k = 1$.

Algorithm 6 COMPUTE-STOCHASTIC-PATH-SCORE($\mathbf{p}_i$) - (Expected Value)

1: $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\} \sim \mathbf{f}(\boldsymbol{\theta})$
2: $\{w_1, \ldots, w_N\} \leftarrow \{w_1, \ldots, w_N\} / \sum_{k=1}^{N} w_k$
3: **for** $k \in \{1, \ldots, N\}$ **do**
4: $\quad \boldsymbol{\tau}_i^\star = \underset{\boldsymbol{\tau}_i}{\text{argmax}} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i), \boldsymbol{\theta}_k)\ x_{ij}$
5: $\quad J_{\mathbf{p}_i}^k = \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^\star(\mathbf{p}_i), \boldsymbol{\theta}_k)\ x_{ij}$
6: **end for**
7: $J_{\mathbf{p}_i} = \sum_{k=1}^{N} w_k\ J_{\mathbf{p}_i}^k$
8: **return** $(J_{\mathbf{p}_i})$

Using sampling, an approximation to the expected-value score for path ($\mathbf{p}_i$) is given by

$$\hat{J}_{\mathbf{p}_i} = \sum_{k=1}^{N} w_k \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^\star(\mathbf{p}_i), \boldsymbol{\theta}_k)\ x_{ij} \right)$$

where the expected value integral is replaced by a summation over the individual weighted samples. Note that within this sampling approximation, the optimal times $\tau_{ij}^\star$ can be deterministically computed for each realization of the uncertainty $\boldsymbol{\theta}_k$ (Alg. 6, line 4), along with the associated path scores (Alg. 6, line 5). These sample path scores can then be used to approximate the stochastic metric, where for the expected-value path score the approximation is given by a sum over weighted score samples (Alg. 6, line 7). In addition to maintaining analytic tractability, another advantage of using sampling is that, although stochastic planning increases the computational complexity of the planning process with respect to the deterministic formulation, the number of samples can be adjusted given the available computational resources. Therefore, there is a trade-off between the accuracy of the approximation versus the amount of time required for the algorithm to run, and real-time convergence guarantees can be preserved by lowering the amount of samples used. In particular, the robust extension to CBBA proposed in this chapter preserves polynomial-time convergence (although the plan time does increase roughly linearly with the number of samples N).

When computing the worst-case path score given the uncertainty in $\boldsymbol{\theta}$, the sampling step can be avoided if intuition about how the uncertainty affects the score function is

Algorithm 7 COMPUTE-STOCHASTIC-PATH-SCORE($\mathbf{p}_i$) - (Worst-Case Value)

1: $\boldsymbol{\tau}_i^\star = \underset{\boldsymbol{\tau}_i}{\text{argmax}} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i), \boldsymbol{\theta}_{worst})\ x_{ij}$

2: $J_{\mathbf{p}_i} = \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^\star(\mathbf{p}_i), \boldsymbol{\theta}_{worst})\ x_{ij}$

3: **return** $(J_{\mathbf{p}_i})$

available. For example, if $\boldsymbol{\theta}$ represents uncertainty in task durations and/or travel times, then $\boldsymbol{\theta}_{worst}$ can be chosen as the longest task durations and the slowest travel times. This is illustrated in Algorithm 7, where the worst-case path score can be analytically computed given the parameter realization $\boldsymbol{\theta}_{worst}$. If the score functions are more complex, and intuitive predictions of how the uncertainty will propagate are hard to make, then the sampling approach used in Algorithm 6 can be employed instead, where line 7 is replaced by $J_{\mathbf{p}_i} = \min_k J_{\mathbf{p}_i}^k$. One issue with using sampling to represent the worst-case performance is that many more samples are required to ensure that low probability catastrophic events are adequately represented. This increases the computational complexity of the algorithm and thus the convergence time (higher N). The field of rare event simulation has addressed this issue by employing smarter sampling methods that focus the sampling process on the low probability zones of the distribution (e.g. importance sampling [11]). These methods could be used to sample efficiently if intuitive predictions on worst-case performance are hard to make.

As discussed in Section 4.1.3, one primary concern with using the distributed CBBA framework is that score functions within CBBA must satisfy a submodularity condition referred to as *diminishing marginal gains* (DMG) in order to ensure algorithm convergence [58]. If the score functions do not satisfy DMG, then the algorithm could lead to cycles between agents' assignments, thus preventing convergence (see Section 4.1.3 for an example). Unfortunately, explicit coupling in the score functions between tasks can often violate this submodularity condition. This is especially true in stochastic scenarios where task scores are coupled through the uncertainty in the planning parameters, and even when the analytic stochastic metrics employed do satisfy submodularity, the use of numerical sampling techniques to compute stochastic path scores could violate DMG, and therefore CBBA is not guaranteed to converge.

As an illustrative example, consider a deterministic time-critical situation with constant-

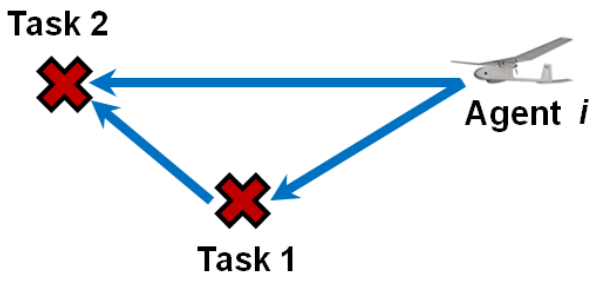


Figure 5-1: Example UAV mission with 1 agent and 2 tasks.

velocity agents, where task scores decrease as a function of time. Figure 5-1 shows an example of this scenario, where an agent has a choice between a far task and a closer task. In this situation, agent i will take longer to reach Task 2 than Task 1, and therefore the score the agent computes for Task 2 is lower than that of Task 1. Thus Task 1 is added to the bundle first. The agent then recomputes a score for Task 2 (which is solely a function of time), and due to the triangle inequality, since the travel distance to Task 2 is now longer than in the previous case without Task 1 in the bundle, the score for Task 2 is necessarily lower than before, satisfying the DMG condition. In uncertain planning environments, however, if the agent velocity is stochastic and sampling methods are employed, it is possible to unluckily select "low-velocity samples" when computing the original score for Task 2, and then "high-velocity samples" when computing the second score for Task 2. Therefore, the algorithm could produce a higher score for Task 2 after Task 1 is added to the bundle, violating the DMG property required by CBBA. As the number of samples goes to infinity, the expected-value score functions may satisfy submodularity, however, given a fixed number of samples, the DMG property is not guaranteed, and therefore Robust CBBA is not guaranteed to converge (as shown in Section 4.1.3, even a violation of DMG by a small value ϵ can cause cycles between agents). In these stochastic settings, designing heuristic approximate score functions that ensure submodularity in bids is a nontrivial exercise, limiting the use of the original CBBA algorithm. This property was identified as part of this work, and it was demonstrated through numerical simulations that this is a crucial issue, leading to cycles within the planner where the algorithm fails to converge [172]. To address this major issue, recent work by Johnson et al. [106] proposed a key algorithmic extension that embeds the DMG condition into the algorithmic framework itself, enabling the use of CBBA with arbitrary (nonsubmodular) score functions. This algorithmic extension was leveraged

within the Robust CBBA framework proposed in this thesis, enabling the use of stochastic metrics while guaranteeing algorithm convergence in distributed stochastic environments. The extension to enable CBBA with nonsubmodular score functions is briefly described in the following section.

5.2.2 CBBA with Nonsubmodular Score Functions

As mentioned in Section 4.1.3, the submodularity requirement imposed by the DMG condition limits the types of objective functions and problem representations allowed within the CBBA framework. For many natural problems of interest, it would be beneficial to allow the use of supermodular objective functions (e.g. clustered tasks, traveling salesman problems, dependent tasks). In recent work [106], we implemented an extension to CBBA proposed by Johnson that enables the algorithm to use score functions that do capture these nonsubmodular effects without having to sacrifice convergence guarantees. A short description of the algorithmic extension is provided here. For further details and proofs of performance and convergence guarantees the reader is referred to [106].

The basic idea behind this nonsubmodular extension to CBBA stems from the key insight that the scores themselves need not be submodular, but the *bids* agents make and share with each other must satisfy DMG. Therefore, as long as the bids shared between agents *appear* to be submodular, CBBA is guaranteed to converge. The proposed algorithmic solution in [106] involves using a *bid warping function*, to adjust the task scores before placing bids, where the resulting bids appear as if they had been made using a submodular score function. This can be accomplished as follows. First, the marginal scores for all available tasks are computed using whatever internal (possibly nonsubmodular) score function the agent wishes to use, $\Delta J_{ij}(\mathbf{p}_i),\ \forall j \in \mathcal{J} \setminus \mathbf{p}_i$. Next, each of these task scores are warped using the following bid warping function,

$$s_{ij} = \min(\Delta J_{ij}(\mathbf{p}_i), y_{ik}), \quad \forall k \in \mathbf{p}_i \tag{5.13}$$

where the y_{ik} values are the current bids that agent i has placed on all the tasks in its bundle, $k \in \mathbf{p}_i$ (recall that if a task k is in agent i's bundle, then y_{ik} holds the corresponding bid value, since agent i believes it is the winner of k and therefore $z_{ik} = i$). The bid warping function in Eq. (5.13) guarantees that each new bid made cannot be greater than the bids

already placed for tasks in the agent's bundle, thus ensuring that all bids satisfy DMG, as if they were made using a submodular score function. Finally, these new bid values, s_{ij}, must be checked against the winning bid list as before, where $h_{ij} = \mathbb{I}(s_{ij} > y_{ij})$ is computed for each available task $j \in \mathcal{J} \setminus \mathbf{p}_i$. A key point is that the best task, however, is selected using the *true* score functions instead of the warped bid values,

$$j^\star = \underset{j \notin \mathbf{p}_i}{\text{argmax}} \ \Delta J_{ij}(\mathbf{p}_i) \ h_{ij} \tag{5.14}$$

This separation between scores and bids allows agents to rank their preferences for tasks using the true (possibly supermodular) score functions yet preserves the convergence guarantees of the original algorithm associated with DMG, thus enabling higher performance. Results are provided in [106] demonstrating the performance improvements achieved in several different planning scenarios by using this novel nonsubmodular extension of CBBA.

5.2.3 Stochastic Bundle Construction

This section describes the stochastic bundle construction process used in Robust CBBA. The full process is summarized in Algorithm 8 and is explained below as follows. As described in Section 4.2, the bundle construction phase of CBBA with time-varying score functions involves each agent iterating over all available tasks $j \in \mathcal{J} \setminus \mathbf{p}_i$, where each task j is inserted into the path at all possible locations n_j to find the optimal position in the path (Algorithm 8, line 3). Using the expected-value stochastic metric, this involves solving the following optimization for each j,

$$J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} = \max_{n_j} \ \mathbb{E}_{\boldsymbol{\theta}} \left\{ \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^\star(\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta}) \ x_{ij} \right\} \tag{5.15}$$

and for the worst-case performance metric, the optimization becomes,

$$J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} = \max_{n_j} \ \min_{\boldsymbol{\theta}} \left\{ \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^\star(\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta}) \ x_{ij} \right\} \tag{5.16}$$

where in both cases the effect of the uncertainty must be propagated throughout the entire path. For the applications considered in this thesis, the uncertainty in the planning process includes variables such as task durations, task positions, and agent velocities. This affects

Algorithm 8 ROBUST-CBBA-BUNDLE-CONSTRUCTION($\mathcal{A}_i, \mathcal{C}_i, \mathcal{J}$)

1: **while** $|\mathbf{p}_i| \leq L_i$ **do**
2: **for** $j \in \mathcal{J} \setminus \mathbf{p}_i$ **do**
3: $J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} \leftarrow \max_{n_j}$ COMPUTE-STOCHASTIC-PATH-SCORE($\mathbf{p}_i \oplus_{n_j} j$)
4: $\Delta J_{ij}(\mathbf{p}_i) = J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} - J_{\mathbf{p}_i}$
5: $s_{ij} = \min(\Delta J_{ij}(\mathbf{p}_i), y_{ik}), \quad \forall k \in \mathbf{p}_i$
6: $h_{ij} = \mathbb{I}(s_{ij} > y_{ij})$
7: **end for**
8: $j^\star = \underset{j \notin \mathbf{p}_i}{\text{argmax}} \ \Delta J_{ij}(\mathbf{p}_i)\ h_{ij}$
9: **if** $(\Delta J_{ij^\star}(\mathbf{p}_i)\ h_{ij^\star} > 0)$ **then**
10: $\mathbf{b}_i \leftarrow (\mathbf{b}_i \oplus_{\text{end}} j^\star)$
11: $\mathbf{p}_i \leftarrow (\mathbf{p}_i \oplus_{n_j^\star} j^\star)$
12: $\boldsymbol{\tau}_i \leftarrow (\boldsymbol{\tau}_i \oplus_{n_j^\star} \tau^\star_{ij^\star}(\mathbf{p}_i \oplus_{n_j^\star} j^\star))$
13: $z_{ij^\star} \leftarrow i$
14: $y_{ij^\star} \leftarrow s_{ij^\star}$
15: **else**
16: **break**
17: **end if**
18: **end while**
19: $\mathcal{A}_i \leftarrow \{\mathbf{b}_i, \mathbf{p}_i, \boldsymbol{\tau}_i\}$
20: $\mathcal{C}_i \leftarrow \{\mathbf{z}_i, \mathbf{y}_i, \mathbf{t}_i\}$
21: **return** $(\mathcal{A}_i, \mathcal{C}_i)$

the times at which tasks will be executed, thus affecting their scores. In particular, inserting a task into the path will impact arrival times for all subsequent tasks, subject to the uncertainty in the system. The stochastic metrics in Eqs. (5.15) and (5.16) capture this coupling between task scores. Note that, as mentioned before in Section 5.2.1, evaluating these stochastic metrics involves finding the optimal task execution times $\tau^\star_{ij}$ for all tasks in the path, for each possible realization of the uncertain parameters $\boldsymbol{\theta}$. Given the extensive coupling and the complications associated with optimizing the task execution times, sampling methods can be used to approximate the stochastic path scores in Eqs. (5.15) and (5.16) instead. In particular, in Algorithm 8, line 3, the function COMPUTE-STOCHASTIC-PATH-SCORE($\mathbf{p}_i \oplus_{n_j} j$) numerically computes the score associated with path ($\mathbf{p}_i \oplus_{n_j} j$) using Algorithm 6 for the expected-value metric and Algorithm 7 for maximization of worst-case performance, as described previously in Section 5.2.1.

The marginal score for each task j is then given by the increase in agent score as a result of adding task j to the path,

$$\Delta J_{ij}(\mathbf{p}_i) = J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} - J_{\mathbf{p}_i} \tag{5.17}$$

as computed by the corresponding stochastic metric (Algorithm 8, line 4). Once the marginal scores $\Delta J_{ij}(\mathbf{p}_i)$ for all available tasks $j \in \mathcal{J} \setminus \mathbf{p}_i$ are computed, the next step is to determine what bid values to make for each task. This involves using the bid warping function described in Section 5.2.2, where each bid is given by

$$s_{ij} = \min(\Delta J_{ij}(\mathbf{p}_i), y_{ik}), \quad \forall k \in \mathbf{p}_i \tag{5.18}$$

as shown in Algorithm 8, line 5. The bids are then checked against the current winning bid list as before, where $h_{ij} = \mathbb{I}(s_{ij} > y_{ij})$ (Algorithm 8, line 6), and the optimal task to add to the bundle is computed using,

$$j^\star = \underset{j \notin \mathbf{p}_i}{\operatorname{argmax}} \ \Delta J_{ij}(\mathbf{p}_i)\ h_{ij} \tag{5.19}$$

where the true internal score values are used to select the highest performing task (Algorithm 8, line 8). Finally, if the score for $j^\star$ is positive, the bundle, path, times, winning agents list, and winning bids list are updated to include the new task,

$$\begin{aligned}
\mathbf{b}_i &\leftarrow (\mathbf{b}_i \oplus_{\text{end}} j^\star) \\
\mathbf{p}_i &\leftarrow (\mathbf{p}_i \oplus_{n_{j^\star}} j^\star) \\
\boldsymbol{\tau}_i &\leftarrow (\boldsymbol{\tau}_i \oplus_{n_{j^\star}} \tau^\star_{ij^\star}(\mathbf{p}_i \oplus_{n_{j^\star}} j^\star)) \\
z_{ij^\star} &= i \\
y_{ij^\star} &= s_{ij^\star}
\end{aligned}$$

This process repeats until no new tasks can be added to the bundle (either because the bundle limit L_i is reached or because no new tasks with positive scores remain).

Note that in this framework, the use of marginal scores within the bundle construction process allows the algorithm to appropriately represent the impact of the uncertainty during every iteration. In other words, even though the bundle is being constructed sequentially, computing the marginal score for tasks requires computing the effect of adding that task on the entire bundle, therefore the coupling between tasks is correctly captured within a consistent framework. Furthermore, even though accounting for uncertainty increases the computational complexity of the planning process with respect to the deterministic problem

formulation, using numerical methods to compute stochastic score functions allows the number of samples to be adjusted given the available computational resources. Therefore, this robust extension to CBBA preserves polynomial-time convergence (although the plan time does increase roughly linearly with the number of samples N).

The sampling methods discussed in this section involve each agent independently sampling from its own local knowledge of the parameter distributions. As such, the number of samples used by each agent can be independently selected given each agent's available computational resources. The accuracy of the internal scores and bids used within Robust CBBA will depend on how good the local situational awareness is for each agent and how many samples are used. The Robust CBBA algorithm, however, is guaranteed to converge even when the situational awareness between agents is different or the bids computed are inaccurate, since it leverages the consensus protocol of the original CBBA algorithm which guarantees convergence even with varying situational awareness between agents [58]. In situations where there is coupling between agent score functions (e.g. through coupled tasks with other agents, or through environment parameters affecting all agents), the distributed CBBA framework can still be used and is still guaranteed to converge, although the performance of the team may decrease if the coupling is not explicitly considered. In these environments, agents would need to share information about the uncertainty to capture the coupling explicitly and improve team performance. Sharing samples between agents would typically require too much communication and thus seems impractical. Current research is considering hyper-parameter consensus methods, where agents can share knowledge about planning parameter distributions (means, variances, etc.) to improve situational awareness and thus increase team performance. In scenarios with explicit coupling, agents might want to share a few representative sample values (e.g. worst-case samples, sigma-points, samples of largest KL-divergence from neighboring agents' distributions, etc.), but deciding what these should be is a nontrivial question which is beyond the scope of this thesis. The next section provides results for example applications validating the performance of the Robust CBBA algorithm proposed in this chapter.

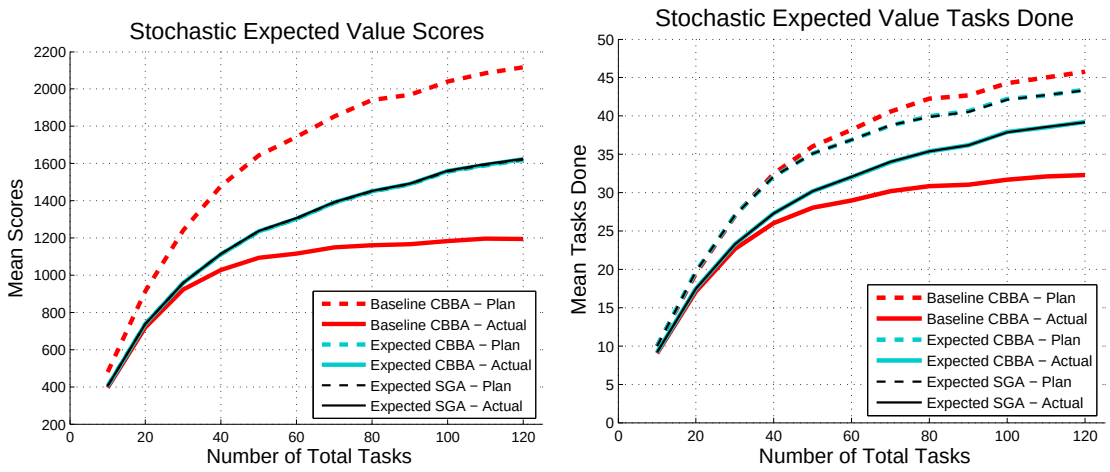
5.3 Example Applications

The distributed Robust CBBA algorithm was implemented in simulation and tested on time-varying UAV missions similar to those described in Section 4.2. The score functions and task and agent parameters were similar to those considered previously in Eq. (4.13), with task rewards defined as,

$$\mathbf{R}_{ij}(\tau_{ij}) = \begin{cases} \mathbf{R}_j \, e^{-\lambda_j \Delta\tau_{ij}}, & t_{j_{\text{start}}} \leq \tau_{ij} \leq t_{j_{\text{end}}} \\ -\mathbf{R}_j, & \text{otherwise} \end{cases}$$

where reward $\mathbf{R}_j$ is obtained if the task is done on time, an exponential discount is applied to the reward to penalize late tasks according to delay $\Delta\tau_{ij} = \max\{0, \tau_{ij} - (t_{j_{\text{start}}} + \bar{t}_{j_{\text{duration}}})\}$ (i.e. delay $\Delta\tau_{ij}$ represents the amount of time in excess of the expected task duration if the task had been started on time), and finally a negative reward of $-\mathbf{R}_j$ is incurred for failing to do the task within the time-window (e.g. representing loss of resources and opportunity cost associated with committing to a task and failing to perform it). In these missions, the length of the actual task durations for some tasks j, $t_{j_{\text{duration}}}$, were considered random variables sampled from a gamma distribution with mean $\bar{t}_{j_{\text{duration}}}$. Three types of tasks were defined: high-reward high-uncertainty tasks, medium-reward tasks with low uncertainty, and low reward tasks but with deterministic service times (same mean duration for all tasks). Two sets of experiments were considered. The first involved using a homogeneous team of UAVs with uncertain velocities (uniform distribution over speed), where all agents had the same statistical properties associated with their stochastic velocities. The second set of experiments considered a heterogeneous UAV mission, where half the team consisted of fast but unpredictable agents (high mean and high variance), and the other half involved slower speed but more predictable agents (lower mean and lower variance), both having uniform distributions on velocities. This section presents results validating the performance of Robust CBBA using the expected-value metric and the worst-case stochastic metric for these two scenarios.

In the first set of experiments, Monte Carlo simulations were performed for stochastic missions of the type described above, where the UAV team consisted of 6 homogeneous agents, and the task space included 50% high-variance tasks, 45% low-variance tasks and 5% deterministic tasks. In these simulations, 3 planning algorithms were used. The first

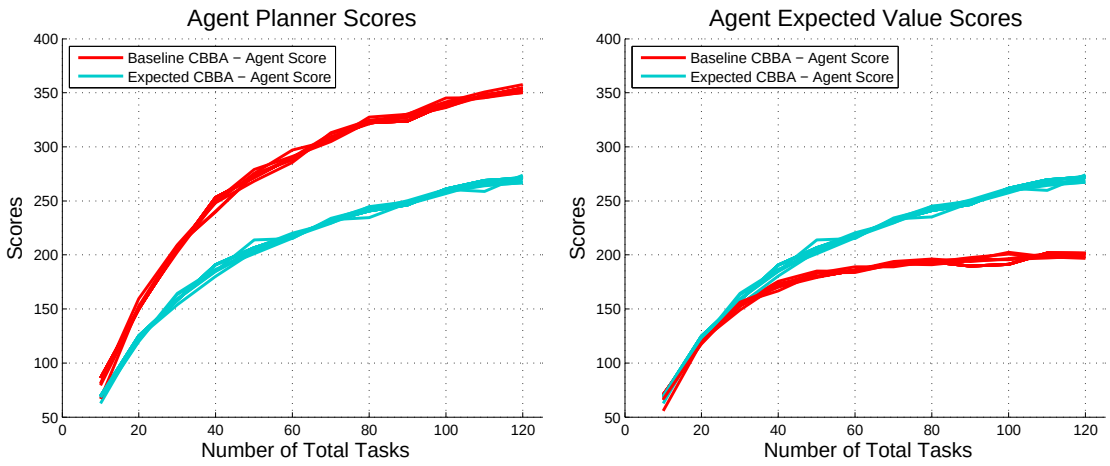


(a) Proposed planner scores vs. actual achieved scores

(b) Proposed planner vs. actual achieved tasks done

Figure 5-2: Monte Carlo simulation results for a stochastic 6 agent mission with homogeneous agents, comparing average mission performance as a function of the number of total available tasks. The plots show the proposed planner output vs. the actual system performance for 3 planning algorithms: Baseline (deterministic) CBBA which uses the mean values of the planning parameters, the distributed Robust Expected-Value CBBA algorithm proposed in this chapter, and a centralized expected-value sequential greedy algorithm (SGA) also optimizing expected-value performance. Figure (a) shows the proposed planner scores (dotted lines) and actual average mission scores (solid lines), and Figure (b) shows the proposed planner tasks vs. the actual number of tasks performed.

algorithm consisted of the baseline deterministic CBBA algorithm using the mean values for all the planning parameters (mean task durations and agent velocities). The second algorithm was the distributed Robust CBBA framework proposed in this chapter using the expected-value metric, where the score was computed numerically as described in Algorithm 6 using $N = 10000$ samples. The third algorithm consisted of a stochastic *centralized* sequential greedy algorithm which also optimized expected-value performance and returned plans for all agents. Figure 5-2 shows the Monte Carlo simulation results comparing the average mission performance of the 6 agent team, as a function of the number of total available tasks in the environment. The plots show the proposed planner output and the actual team performance for the 3 planning algorithms described above: Baseline (deterministic) CBBA (red), the distributed Robust Expected-Value CBBA algorithm proposed in this chapter (cyan), and the centralized expected-value sequential greedy algorithm (SGA) also optimizing expected-value performance (black). Figure 5-2(a) shows the proposed planner scores (dotted lines) and the actual average mission scores achieved by the team (solid

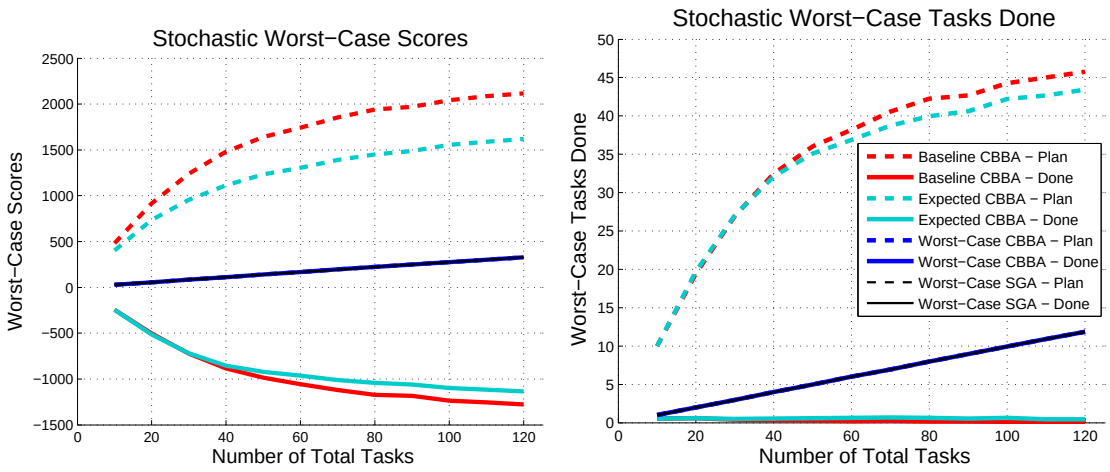


(a) Proposed planner scores for individual agents

(b) Actual average scores for individual agents

Figure 5-3: Monte Carlo simulation results showing individual agent contributions for a stochastic 6 agent mission with homogeneous agents. The results compare the agents' average performance using Baseline (deterministic) CBBA and the distributed Robust Expected-Value CBBA algorithm proposed in this chapter. The plots show the proposed planner scores and actual average scores for each individual agent as a function of the number of tasks.

lines). Figure 5-2(b) shows the number of tasks proposed by the planner (dotted lines) and the actual number of tasks performed on average by the team (solid lines). As seen in Figure 5-2, the Robust Expected-Value CBBA algorithm achieves higher mean mission performance than Baseline CBBA, since the deterministic baseline planner fails to capture the coupling between tasks, i.e. the fact that delays caused by longer than expected task durations impact the scores for the remaining tasks in the path, leading to poor performance. Robust CBBA is able to hedge against this uncertainty to obtain an improved plan, leading to higher average scores and a greater number of completed tasks. It is also worth noting that the distributed Robust CBBA algorithm achieves similar performance to the centralized stochastic sequential greedy algorithm, validating the distributed approach. A further interesting point is that the proposed planner scores and the number of proposed tasks (dotted lines) using the deterministic Baseline CBBA algorithm are higher than using Robust CBBA, even though the actual average mission scores and tasks done are lower. This is because the deterministic CBBA algorithm tries to squeeze tasks into the current path based on expected values of the planning parameters. Therefore it fails to capture that taking longer than expected on early tasks will likely cause later task execution times to be outside the task time-windows, thus resulting in task failures. As a result, the deterministic

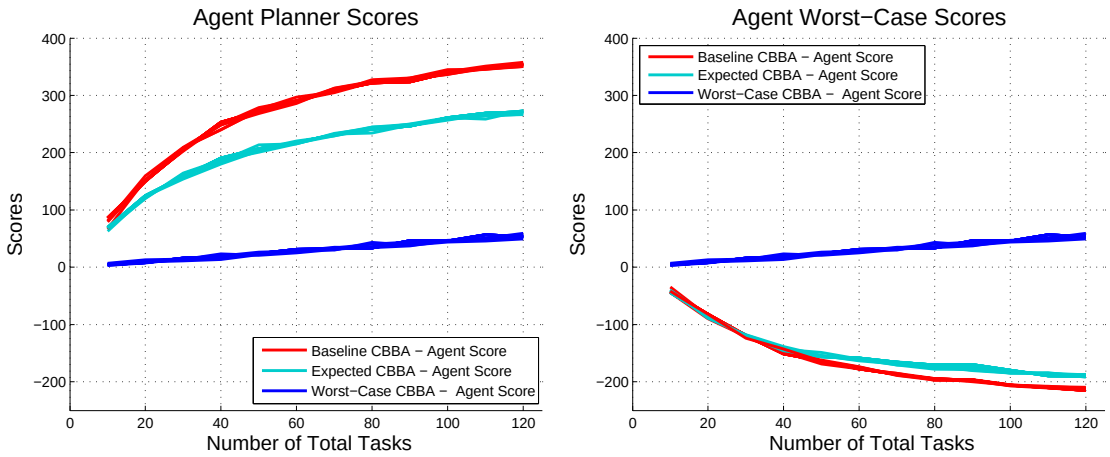


(a) Proposed planner scores vs. actual achieved scores

(b) Proposed planner vs. actual achieved tasks done

Figure 5-4: Monte Carlo simulation results for a stochastic 6 agent mission with homogeneous agents, comparing worst-case mission performance as a function of the number of total available tasks. The plots show the proposed planner output vs. the actual system performance for 4 planning algorithms: Baseline (deterministic) CBBA, the Robust Expected-Value CBBA algorithm (optimizing average performance, not worst-case), the proposed Robust Worst-Case CBBA algorithm, and a centralized worst-case sequential greedy algorithm (SGA) explicitly optimizing worst-case team performance. Figure (a) shows the proposed planner scores (dotted lines) and actual worst-case mission scores (solid lines), and Figure (b) shows the proposed planner tasks vs. the actual number of tasks performed in the worst case.

planner assigns more tasks overall, but often fails to execute all the tasks in its path since some of these tasks are pushed outside their time-windows. Robust CBBA, on the other hand, accounts for these potential delays and will not add tasks to the path if they impact the arrival times of high value tasks later in the path. As a result, Robust CBBA creates a more conservative plan with fewer assigned tasks and more buffers between them, but also succeeds in correctly executing more of these tasks, thus achieving higher scores. Figure 5-3 shows the score performance for each individual agent using the baseline CBBA algorithm and the Robust Expected-Value CBBA algorithm. Figure 5-3(a) shows the proposed planner scores for each agent, and Figure 5-3(b) shows the average mission scores achieved by each agent. As seen in the plots, all agents propose very similar plans and achieve similar average scores, since the team is homogeneous, demonstrating that the CBBA algorithm distributes the tasks evenly amongst the agents. Furthermore, each agent's actual score is higher using the Robust CBBA algorithm (Figure 5-3(b)), and the planner predictions are

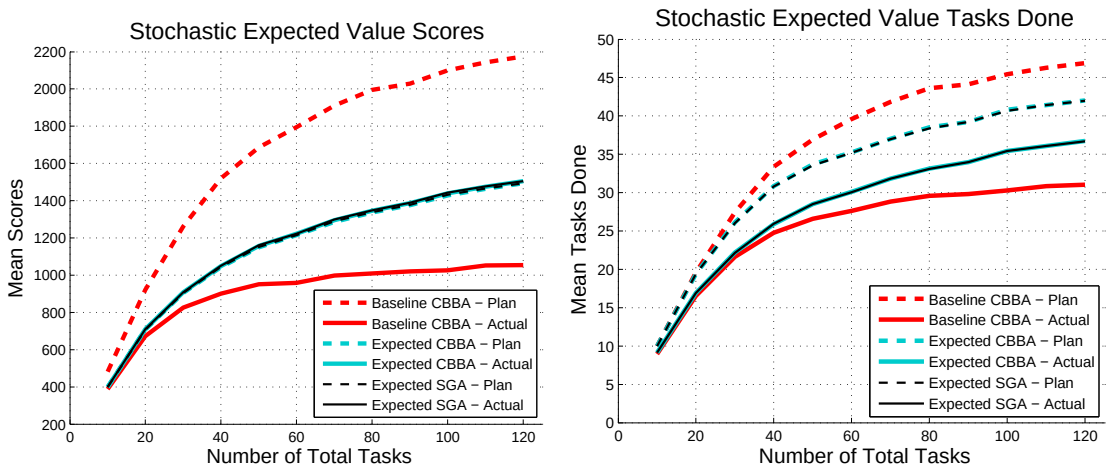


(a) Proposed planner scores for individual agents (b) Actual worst-case scores for individual agents

Figure 5-5: Monte Carlo simulation results showing individual agent contributions for a stochastic 6 agent mission with homogeneous agents. The results compare the agents' worst-case performance using Baseline (deterministic) CBBA, Robust Expected-Value CBBA, and the proposed Robust Worst-Case CBBA algorithm. The plots show the proposed planner scores and actual worst-case scores for each individual agent as a function of the number of tasks.

more accurate as well (proposed and actual scores using Robust CBBA are much closer than those using deterministic baseline CBBA).

In missions with low tolerance to failure, the worst-case performance is often of more interest than the average mission performance. To compare the performance of the different planning algorithms in these settings, the experiments described above were repeated, this time analyzing the worst-case mission performance. In these simulations, 4 planning algorithms were used. The first algorithm consisted of the baseline deterministic CBBA algorithm using the mean values for all the planning parameters. The second algorithm was Robust CBBA using the expected-value metric as before (optimizing average performance, not worst-case). The third algorithm was Robust CBBA, but using the worst-case stochastic metric as described in Algorithm 7, and the fourth consisted of a stochastic centralized sequential greedy algorithm (SGA) explicitly optimizing worst-case team performance as well. Figure 5-4 shows the Monte Carlo simulation results comparing the worst-case mission performance of the 6 agent team, as a function of the number of total available tasks in the environment. The plots show the proposed planner output and the actual team performance for the 4 planning algorithms described above: Baseline (deterministic) CBBA (red), Robust Expected-Value CBBA (cyan), the proposed Robust Worst-Case CBBA algorithm

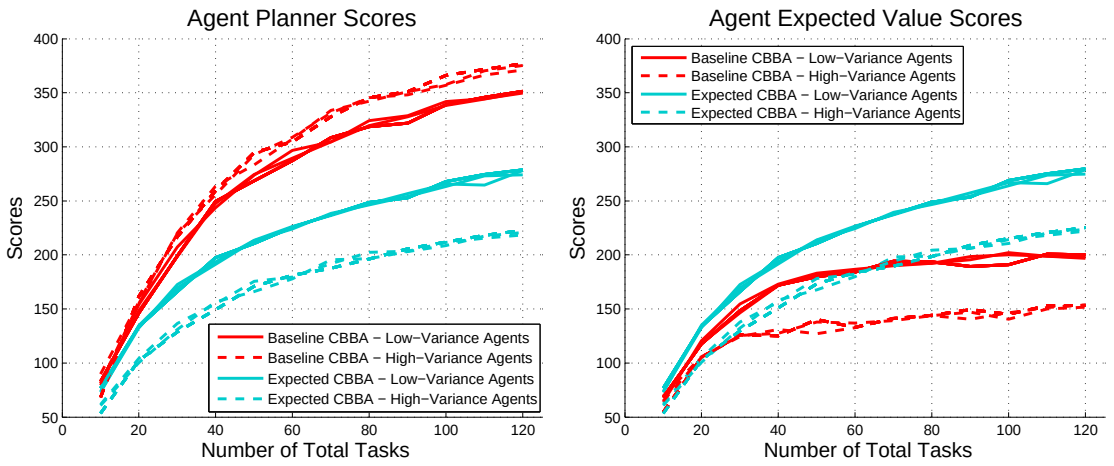


(a) Proposed planner scores vs. actual achieved scores

(b) Proposed planner vs. actual achieved tasks done

Figure 5-6: Monte Carlo simulation results for a stochastic 6 agent mission with heterogeneous agents, comparing average mission performance as a function of the number of total available tasks. The plots show the proposed planner output vs. the actual system performance for 3 planning algorithms: Baseline (deterministic) CBBA which uses the mean values of the planning parameters, the distributed Robust Expected-Value CBBA algorithm proposed in this chapter, and a centralized expected-value sequential greedy algorithm (SGA) also optimizing expected-value performance. Figure (a) shows the proposed planner scores (dotted lines) and actual average mission scores (solid lines), and Figure (b) shows the proposed planner tasks vs. the actual number of tasks performed.

(blue), and the centralized worst-case sequential greedy algorithm (black). Figure 5-4(a) shows the proposed planner scores (dotted lines) and the actual worst-case mission scores achieved by the team (solid lines). Figure 5-4(b) shows the number of tasks proposed by the planner (dotted lines) and the actual number of tasks performed in the worst case by the team (solid lines). As seen in Figure 5-4, the Robust Worst-Case CBBA algorithm achieves significantly higher mission performance in this worst-case than Baseline CBBA or Robust Expected-Value CBBA, since these planners fail to capture the fact that really long task execution times may occur for the stochastic tasks. Robust Worst-Case CBBA, on the other hand, is able to explicitly maximize the worst-case performance by accounting for the extreme values of the parameter distributions. The proposed planner scores and the number of proposed tasks (dotted lines) were highest for the deterministic Baseline CBBA algorithm, and the achieved performance was the lowest (solid lines), since this deterministic algorithm did not account for any uncertainty in the planning parameters. The Robust Expected-Value CBBA algorithm achieved slightly better planner predictions and

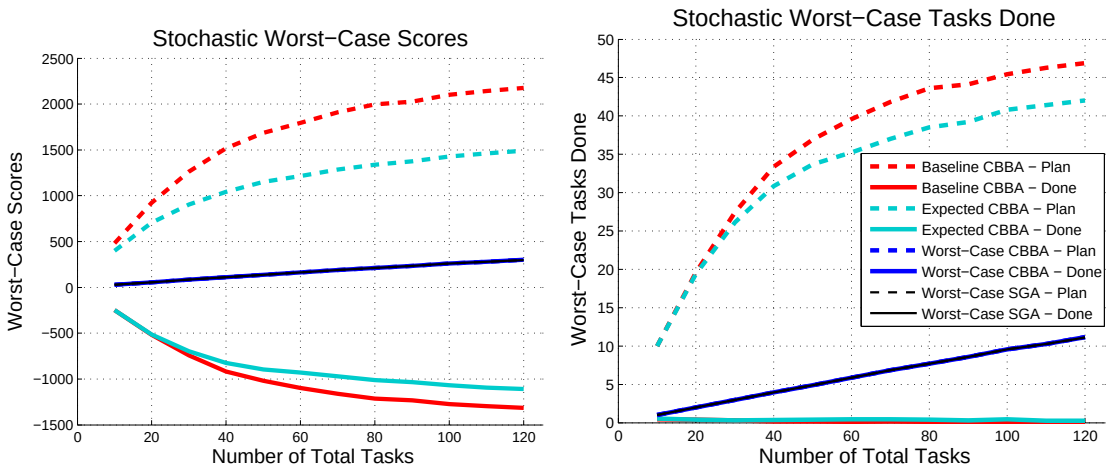


(a) Proposed planner scores for individual agents

(b) Actual average scores for individual agents

Figure 5-7: Monte Carlo simulation results showing individual agent contributions for a stochastic 6 agent mission with heterogeneous agents. The results compare the agents' average performance using Baseline (deterministic) CBBA and the distributed Robust Expected-Value CBBA algorithm proposed in this chapter. The plots show the proposed planner scores and actual average scores for each individual agent as a function of the number of tasks.

slightly higher performance than the deterministic case, since the coupling in parameter uncertainty was being captured, but the worst-case performance was still quite low since the algorithm was optimizing the average team performance. The Robust Worst-Case CBBA algorithm was able to explicitly optimize the worst-case scenarios, leading to accurate planner predictions and higher worst-case team performance. As before, the distributed Robust Worst-Case CBBA algorithm achieved similar performance to the centralized stochastic sequential greedy algorithm, validating the distributed approach. Figure 5-5 shows the score performance for each individual agent using the baseline CBBA algorithm, the Robust Expected-Value CBBA algorithm, and the Robust Worst-Case CBBA algorithm. Figure 5-5(a) shows the proposed planner scores for each agent, and Figure 5-5(b) shows the worst-case mission scores achieved by each agent. Again, as seen in the plots, all agents propose very similar plans and achieve similar worst-case scores, since the team is homogeneous, demonstrating that CBBA distributes the tasks evenly amongst the agents. Furthermore, each agent's actual score is highest using the Robust Worst-Case CBBA algorithm (Figure 5-5(b)), and the planner predictions are more accurate as well (proposed and actual scores using Robust Worst-Case CBBA are identical, which is not the case for the other two algorithms).

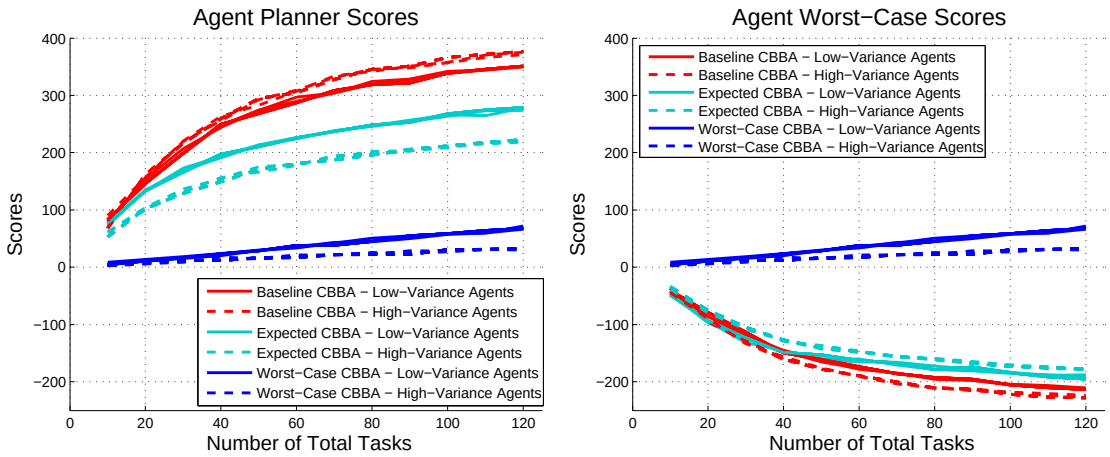


(a) Proposed planner scores vs. actual achieved scores

(b) Proposed planner vs. actual achieved tasks done

Figure 5-8: Monte Carlo simulation results for a stochastic 6 agent mission with heterogeneous agents, comparing worst-case mission performance as a function of the number of total available tasks. The plots show the proposed planner output vs. the actual system performance for 4 planning algorithms: Baseline (deterministic) CBBA, the Robust Expected-Value CBBA algorithm (optimizing average performance, not worst-case), the proposed Robust Worst-Case CBBA algorithm, and a centralized worst-case sequential greedy algorithm (SGA) explicitly optimizing worst-case team performance. Figure (a) shows the proposed planner scores (dotted lines) and actual worst-case mission scores (solid lines), and Figure (b) shows the proposed planner tasks vs. the actual number of tasks performed in the worst case.

In the second set of experiments, the simulation trials described above were repeated, but using a heterogeneous UAV team instead, where half of the agents consisted of fast but unpredictable vehicles (high mean and high variance), and the other half involved slower speed but more predictable agents (lower mean and lower variance), both having uniform distributions on velocities. Figures 5-6 and 5-7 show the average mission performance for the heterogeneous team, and Figures 5-8 and 5-9 show the worst-case mission performance. As seen in Figures 5-6 and 5-8, the results obtained for the heterogeneous team were very similar to those obtained for the homogeneous team scenarios. The breakdown of scores for each agent, however, were different in these heterogeneous cases, since the agents had different planning parameters and statistical properties. In Figure 5-7(a), the results show that the scores proposed by the deterministic planner were higher for the high-variance agents than the low-variance agents, since the high-variance agents had higher mean velocities as well. The actual average performance of the high-variance agents, however, was worse than that

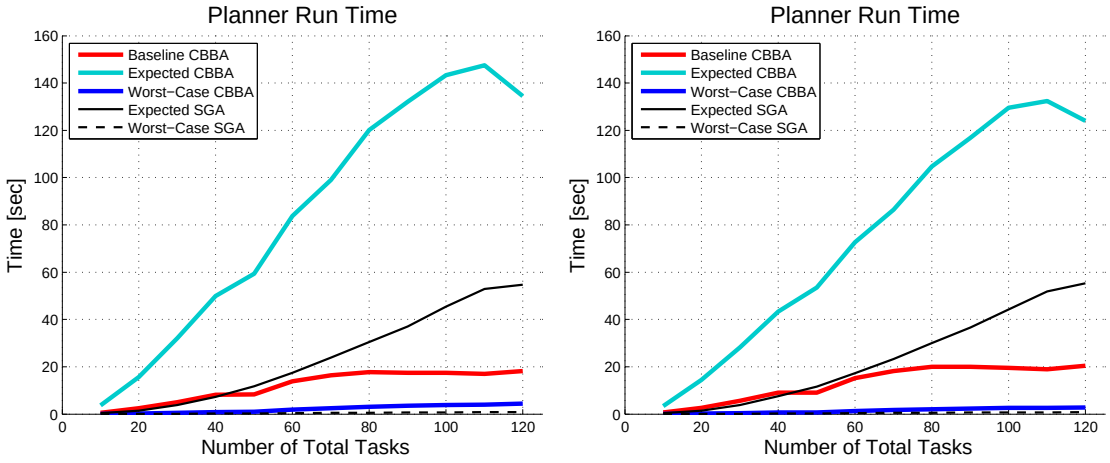


(a) Proposed planner scores for individual agents (b) Actual worst-case scores for individual agents

Figure 5-9: Monte Carlo simulation results showing individual agent contributions for a stochastic 6 agent mission with heterogeneous agents. The results compare the agents' worst-case performance using Baseline (deterministic) CBBA, Robust Expected-Value CBBA, and the proposed Robust Worst-Case CBBA algorithm. The plots show the proposed planner scores and actual worst-case scores for each individual agent as a function of the number of tasks.

of the low-variance agents (see Figure 5-7(b)), since the higher variance in agent velocities coupled into the task scores and propagated through the agents' paths, negatively impacting the average performance (e.g. higher probabilities of being late for tasks, or being outside the task time-windows). The Robust Expected-Value CBBA planner, on the other hand, was able to predict this coupling effect, and therefore proposed more conservative plans (with lower scores) for the high-variance agents. This effect is seen in Figure 5-9 as well comparing the worst-case performance for each agent. Here the Robust Worst-Case CBBA planner accounts for the fact that the worst-case velocities achieved for the high-variance agents are slower than for the low-variance agents (even though the mean velocities are faster), and therefore more conservative plans are proposed for the high-variance agents (with correspondingly lower scores).

Finally, comparisons of planner run time for the different stochastic planning algorithms are provided in Figure 5-10, including the Baseline (deterministic) CBBA algorithm, the Robust Expected-Value CBBA algorithm, the Robust Worst-Case CBBA algorithm, the centralized Expected-Value sequential greedy algorithm, and the centralized Worst-Case sequential greedy algorithm. As shown in the plots, the worst-case robust algorithms perform the fastest. This is because the worst-case stochastic metric employed in these scenarios



(a) Planner run time (homogeneous case)

(b) Planner run time (heterogeneous case)

Figure 5-10: Comparison of planner run time for the different stochastic planning algorithms, including Baseline (deterministic) CBBA, Robust Expected-Value CBBA, Robust Worst-Case CBBA, centralized Expected-Value SGA, and centralized Worst-Case SGA. All computations were done in single threads, programmed in MATLAB, on an Alienware computer with an Intel Core i7 processor and 12 GB RAM. These results show the total computation time (for all agents combined).

exploited domain knowledge to perform the worst-case path score computations analytically instead of using samples, thus lowering the computation time. The baseline CBBA algorithm run time is higher than for the worst-case algorithms because the number of tasks selected (and explored) during the planning process for the worst-case algorithms is lower, since these algorithms are very conservative, and therefore the number of iterations during the task selection process is also lower. The expected-value algorithms use sampling to represent the stochastic path scores and are thus the most computationally intensive, however, the algorithm run time is still reasonable and scales roughly linearly given the number of tasks in the environment (also scales roughly linearly with the number of agents). The reduction in run time for the Robust Expected-Value CBBA algorithm which occurs between 100 and 120 tasks is associated with the fact that agents have more choices and are therefore less likely to conflict, leading to fewer iterations of CBBA. In general, the distributed CBBA run time is a function of the number of tasks explored by each agent, the number of conflicts between agents (and thus iterations of CBBA), and the amount of time required to compute each path score within each agent's bundle optimization process. Both centralized sequential greedy algorithms were faster than their distributed counterparts, since more iterations are required to deconflict plans using the distributed CBBA algorithms than in

the centralized case. These experiments demonstrate the functionality and applicability of the algorithms proposed in this thesis, and the results in Figure 5-10 serve to illustrate the relative run time of the different algorithms with respect to each other. As a disclaimer, all computations were done in single threads, programmed in MATLAB, on an Alienware computer with an Intel Core i7 processor and 12 GB RAM, and these results show the total computation time for all agents combined. In practical real-time implementations, the computation would be distributed/parallelized over several computers and written in a more efficient language than MATLAB (e.g. C++), therefore the true computation time would be reduced by at least a factor of N (since each agent would compute its own plans), and would possibly be even faster given a more efficient programming language. Of course, the distributed implementation would also have to account for communication speed between agents, which, depending on the application at hand, may introduce further delays.

This chapter extended the deterministic CBBA with time-varying score functions algorithm to optimize performance in stochastic environments, providing a distributed real-time framework which can leverage different stochastic metrics to hedge against parameter uncertainty given probabilistic agent and task models. There are several key features of this Robust CBBA algorithm. Firstly, the algorithm has the ability to handle the nontrivial coupling between the stochastic metrics and the decision variables, associated with optimizing the task execution times, using sampling methods. Secondly, the algorithm leverages the convergence guarantees of CBBA under differing situational awareness [58] to allow agents to individually sample their uncertainty when building their bundles. The algorithm also leverages a recent CBBA extension to allow nonsubmodular score functions within the planning framework [106] to enable the use of sampling methods within the bundle construction process. Using sampling to approximate stochastic metrics allows Robust CBBA to maintain analytic and computational tractability, providing polynomial-time convergence guarantees. And finally, within the distributed framework, agents can independently select the number of samples used within the scoring functions given their own available resources, leading to a tradeoff between solution quality and algorithm convergence time which can be optimized given the specific mission requirements.

The Robust CBBA framework was demonstrated using the expected-value metric, achieving improvements in average team performance, but with fairly large variability associated with mission scores. For missions where stronger performance guarantees are required, the

Robust CBBA framework can use a worst-case stochastic metric to optimize the lowest achievable mission scores, where the score predicted by the planner provides a lower bound on the achievable mission performance. However, for most scenarios of interest this worst-case approach is too conservative, and an intermediate approach which can mitigate the conservatism of the solution while still providing performance guarantees would be more desirable. The next chapter describes a chance-constrained stochastic extension to CBBA that allows finer control over the degree of conservatism used in the planner, creating robust plans that optimize performance within allowable risk thresholds.

Chapter 6

Distributed Risk-Aware Planning in Uncertain Domains

The previous chapter discussed how to extend CBBA to account for stochastic environments by optimizing expected-value plans and maximizing worst-case mission performance. The chance-constrained formulation [45, 66, 157], which guarantees that the global mission performance will be at least as good as the proposed plan value within a certain allowable risk threshold, provides more flexibility over the conservatism of the solution. Unfortunately, as described in the previous chapter, the chance-constrained formulation couples agent assignments through a joint probability constraint, making distributed implementation difficult. This chapter revisits the chance-constrained formulation, and proposes an extension to CBBA that allows an approximation of the chance-constrained problem to be solved in a distributed fashion. The following sections provide more details about the distributed problem formulation and the proposed approach.

6.1 Distributed Chance-Constrained Problem Formulation

Consider the chance-constrained problem formulation defined in Eq. (5.9) and repeated in Eq. (6.1) for convenience. An intuitive illustration of the chance-constrained metric is provided in Figure 6-1. The problem formulation in Eq. (6.1) is the chance-constrained extension of the deterministic problem formulation considered in Eq. (2.6) with all the simplifying assumptions described previously in Section 2.1.3. Not addressed in this thesis is the more general chance-constrained case involving probabilistic coupled constraints,

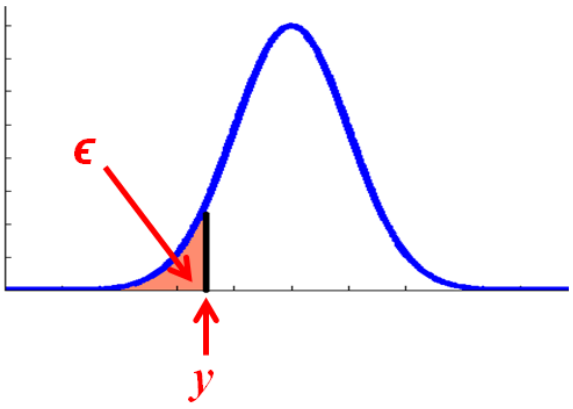


Figure 6-1: Illustration of the chance-constrained metric, which involves optimizing the worst-case performance within an allowable risk threshold ϵ. Here the chance-constrained score is given by y, and can be obtained by integrating and removing the low-probability zone, where the area under the curve is of size ϵ.

which are typically much harder to deal with [157]. In the general chance-constrained formulation, this requires ensuring that constraints are satisfied within a given probability, e.g. $\mathbb{P}_{\boldsymbol{\theta}}\left(\mathbf{G}(\mathbf{x}, \boldsymbol{\tau}, \boldsymbol{\theta}) \leq \mathbf{b}\right) > \alpha$, for some specified probability bound α.

$$
\begin{aligned}
\max_{\mathbf{x},\boldsymbol{\tau}} \quad & y \\
\text{s.t.} \quad & \mathbb{P}_{\boldsymbol{\theta}}\left\{\sum_{i=1}^{N_a}\left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij}\right) > y\right\} \geq 1-\epsilon \\
& \sum_{j=1}^{N_t} x_{ij} \leq L_i, \quad \forall i \in \mathcal{I} \\
& \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\
& x_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \\
& \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J}
\end{aligned}
\tag{6.1}
$$

The major difficulty with solving Eq. (6.1) in a distributed manner is that the agents' assignments are now coupled through a joint probability constraint (2^{nd} line of Eq. (6.1)). As mentioned previously in Chapter 3, any joint constraint between the agents requires designing adequate consensus protocols within the distributed algorithm. Therefore extending CBBA to solve the above chance-constrained problem would require modifying the consensus process to ensure that the probabilistic constraint is met. Since this is a nontriv-

ial endeavor [220, 221], an alternate strategy is to formulate a distributed approximation to the above chance-constrained formulation. This is accomplished by decomposing the centralized problem of Eq. (6.1) into distributable sub-problems of the following form,

$$\begin{aligned}
\max_{\mathbf{x}_i, \boldsymbol{\tau}_i} \quad & y_i \\
\text{s.t.} \quad & \mathbb{P}_{\boldsymbol{\theta}} \left\{ \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij} \right) > y_i \right\} \geq 1 - \epsilon_i, \\
& \sum_{j=1}^{N_t} x_{ij} \leq L_i, \\
& \sum_{i=1}^{N_a} x_{ij} \leq 1, \quad \forall j \in \mathcal{J} \\
& x_{ij} \in \{0, 1\}, \quad \forall j \in \mathcal{J} \\
& \tau_{ij} \in \{\mathbb{R}^+ \cup \emptyset\}, \quad \forall j \in \mathcal{J}
\end{aligned} \tag{6.2}$$

where the chance-constrained mission score, y, is approximated by a sum over the individual agent chance-constrained scores y_i (i.e. $\hat{y} = \sum_{i=1}^{N_a} y_i$ is an approximation of the original centralized chance-constrained mission score y). Note that in these individual agent chance-constrained sub-problems, the allowable risk thresholds for each agent are given by ϵ_i, which are in general not equal to the mission risk ϵ, and the values selected for these individual risk allotments impact the accuracy of the approximation given by $\hat{y}$. It is therefore essential to select proper values of the individual risk thresholds, ϵ_i, such that solving the distributed chance-constrained optimizations in Eq. (6.2) gives a good approximation to the centralized chance-constrained optimization (Eq. (6.1)). More details on how to select these risk values are provided in Section 6.2.

Using this distributed chance-constrained approximation within the CBBA framework, each agent i can solve its own chance-constrained optimization to maximize y_i subject to its individual risk threshold ϵ_i, while ensuring, through communication with other agents, that the joint constraint for a non-conflicting solution remains satisfied (i.e. the constraint specifying that each task can be assigned to at most one agent, as described previously in Eq. (2.3)). The planner score for the team predicted by CBBA is then given by the sum over the local chance-constrained agent scores. The challenge with solving this distributed chance-constrained optimization involves developing analytic expressions that relate each

agent's risk ϵ_i to the global risk ϵ within a theoretically sound framework, to make the distributed approximation $\hat{y}$ as close as possible to the original centralized score y of Eq. (6.1). Expressions for choosing each agent's risks are typically analytically intractable and problem specific, so the challenge lies in developing good approximations to relate the global and local risk thresholds, which is the subject of the next section.

6.2 Allocating Agent Risks in Distributed Chance-Constrained Planning

As mentioned previously, a key question in this problem is characterizing how the agent risks used in the distributed chance-constrained approximation relate to the global mission risk. Revisiting the centralized problem statement in Eq. (6.1), the global mission score is maximized by solving,

$$\begin{aligned} \max_{\mathbf{x},\boldsymbol{\tau}} \quad & y \\ \text{s.t.} \quad & \mathbb{P}_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta}) \; x_{ij} \right) > y \right\} \geq 1 - \epsilon \end{aligned} \tag{6.3}$$

whereas in the distributed chance-constrained optimization of Eq. (6.2), each agent's score is maximized by solving,

$$\begin{aligned} \max_{\mathbf{x}_i,\boldsymbol{\tau}_i} \quad & y_i \\ \text{s.t.} \quad & \mathbb{P}_{\boldsymbol{\theta}} \left\{ \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta}) \; x_{ij} \right) > y_i \right\} \geq 1 - \epsilon_i \end{aligned} \tag{6.4}$$

and the global mission score is approximated by $\hat{y} = \sum_{i=1}^{N_a} y_i$. The following describes how these two optimizations can be related.

To simplify the notation, we first define new random variables $\mathbf{z}$ and $\mathbf{z}_i$, $\forall i$, representing the uncertain mission score and agent scores respectively, which are given by,

$$\mathbf{z} \;=\; \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta}) \; x_{ij} \right)$$

$$\mathbf{z}_i \quad = \quad \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)), \boldsymbol{\theta})\ x_{ij}, \quad \forall i \in \mathcal{I}$$

where $\mathbf{z} = \sum_{i=1}^{N_a} \mathbf{z}_i$. Fixing values for the assignments $\mathbf{x}$ and the task execution times $\boldsymbol{\tau}$, the distribution (PDF) of the mission score $\mathbf{f}(\mathbf{z})$ can be derived from the joint distribution of the uncertainty parameters, $\mathbf{f}(\boldsymbol{\theta})$, for each given plan specified by $\mathbf{x}$ and $\boldsymbol{\tau}$ (note that this distribution cannot usually be computed in closed form and numerical approximations are often employed as described in the next section). Given a plan $\mathbf{x}$ and $\boldsymbol{\tau}$ and the associated random mission score $\mathbf{z}$, Eq. (6.3) can be written as,

$$\begin{aligned} \max \quad & y \\ \text{s.t.} \quad & \mathbb{P}_{\mathbf{z}}\{\mathbf{z} > y\} \geq 1 - \epsilon \end{aligned}$$

or equivalently,

$$\begin{aligned} \max \quad & y \\ \text{s.t.} \quad & \mathbb{P}_{\mathbf{z}}\{\mathbf{z} \leq y\} \leq \epsilon \end{aligned}$$

And, using the cumulative distribution function (CDF) of the random variable $\mathbf{z}$, $\mathbf{F}_{\mathbf{z}}(y) = \mathbb{P}_{\mathbf{z}}\{\mathbf{z} \leq y\}$, the optimization in Eq. (6.3) can be re-written as,

$$\begin{aligned} \max \quad & y \\ \text{s.t.} \quad & \mathbf{F}_{\mathbf{z}}(y) \leq \epsilon \end{aligned} \tag{6.5}$$

In similar fashion, given plans for each agent i, $\mathbf{x}_i$ and $\boldsymbol{\tau}_i$, and the associated random agent scores, $\mathbf{z}_i$, Eq. (6.4) for each agent can be written as,

$$\begin{aligned} \max \quad & y_i \\ \text{s.t.} \quad & \mathbf{F}_{\mathbf{z}_i}(y_i) \leq \epsilon_i \end{aligned} \tag{6.6}$$

where $\mathbf{F}_{\mathbf{z}_i}(y_i) = \mathbb{P}_{\mathbf{z}_i}\{\mathbf{z}_i \leq y_i\}$ is the CDF of each agent's score.

Given Eqs. (6.5) and (6.6) which describe the centralized and distributed chance-constrained optimizations, the goal is to ensure that solving Eq. (6.6) for all agents i adequately rep-

resents solving the global optimization of Eq. (6.5). In the following derivation, we will show that a constraint can be imposed on the agent risks ϵ_i given a global mission risk threshold ϵ, and that adding this constraint to the distributed chance-constrained optimization of Eq. (6.6) ensures that the solution to the distributed approximation will satisfy the chance-constraint imposed in the centralized optimization of Eq. (6.5).

To begin the derivation, we will start from Eq. (6.5) and write successively more constrained problems until an equivalence with Eq. (6.6) is reached. First, we assume that CDFs for the mission and agent score distributions are invertible, i.e. $\mathbf{F}_{\mathbf{z}}^{-1}(\cdot)$ and $\mathbf{F}_{\mathbf{z}_i}^{-1}(\cdot)$ exist (e.g. for continuous random variables), and leveraging the fact that CDFs are monotonic functions, Eq. (6.5) can be re-written as,

$$\begin{aligned} \max \quad & y \\ \text{s.t.} \quad & y \leq \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon) \end{aligned}$$

Next, we consider a more constrained version of this optimization given by,

$$\begin{aligned} \max \quad & \sum_{i=1}^{N_a} y_i \\ \text{s.t.} \quad & y \leq \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon) \\ & \sum_{i=1}^{N_a} y_i \leq y \end{aligned}$$

where the decision variables y_i must be maximized subject to the constraint $\sum_{i=1}^{N_a} y_i \leq y$. Since this is a more constrained optimization than that of Eq. (6.5), a score obtained by summing over the values of y_i that optimize this problem is guaranteed to satisfy the constraint specified in Eq. (6.5). In similar fashion, we introduce a new constraint next using variables $\mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i)$, producing a further constrained optimization,

$$\begin{aligned} \max \quad & \sum_{i=1}^{N_a} y_i \\ \text{s.t.} \quad & y \leq \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon) \\ & \sum_{i=1}^{N_a} \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i) \leq y \end{aligned}$$

$$\sum_{i=1}^{N_a} y_i \leq \sum_{i=1}^{N_a} \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i)$$

Finally, we note that when $y_i \leq \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i), \forall i$, the last constraint, $\sum_{i=1}^{N_a} y_i \leq \sum_{i=1}^{N_a} \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i)$, is always satisfied (although the converse is not true), leading to a further constrained optimization

$$\begin{aligned} \max \quad & \sum_{i=1}^{N_a} y_i \\ \text{s.t.} \quad & y \leq \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon) \\ & \sum_{i=1}^{N_a} \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i) \leq y \\ & y_i \leq \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i), \quad \forall i \end{aligned}$$

Rearranging the equations in the above optimization and merging constraints gives the following optimization problem,

$$\begin{aligned} \max \quad & \sum_{i=1}^{N_a} y_i \\ \text{s.t.} \quad & y_i \leq \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i), \quad \forall i \\ & \sum_{i=1}^{N_a} \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i) \leq \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon) \end{aligned} \tag{6.7}$$

which looks very similar to the distributed chance-constrained approximation specified in Eq. (6.6), but with an additional constraint imposed on the agent risk thresholds. Solving the optimization in Eq. (6.7) ensures satisfaction of the centralized chance-constraint in Eq. (6.5), since Eq. (6.7) was derived as a more constrained version of the original optimization in Eq. (6.5) (in other words, $\mathbf{F}_{\mathbf{z}}(\hat{y}) \leq \epsilon$ is guaranteed to be true given $\hat{y} = \sum_{i=1}^{N_a} y_i$ where the decision variables y_i are given by solving the optimization in Eq. (6.7)). Therefore, setting the agent risks ϵ_i such that the constraint in Eq. (6.7) is met, ensures that, by solving the distributed approximation, the centralized chance-constraint will be satisfied.

Note that the above set of optimizations are maximized when $y_i = \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i)$, $y = \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)$, and $y = \sum_{i=1}^{N_a} y_i$. Therefore, for a given value of ϵ, and for a given plan $\mathbf{x}$ and $\boldsymbol{\tau}$ (with $\mathbf{x}_i$ and $\boldsymbol{\tau}_i$ specified for all agents), with associated mission and agent distributions $\mathbf{f}(\mathbf{z})$ and

$\mathbf{f}(\mathbf{z}_i), \forall i$, the constraint that specifies how to set agent risks given mission risk is written as,

$$\sum_{i=1}^{N_a} \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i) = \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon) \tag{6.8}$$

The main practical issue associated with using Eq. (6.8) to allocate agent risks within a planning framework is that the agent score distributions $\mathbf{f}(\mathbf{z}_i)$ and the mission score distribution $\mathbf{f}(\mathbf{z})$ are not usually available *a priori*. Therefore, using Eq. (6.8) within the planner involves approximating these score distributions with heuristics. A second challenge is that, even if the distributions $\mathbf{f}(\mathbf{z}_i), \forall i$ and $\mathbf{f}(\mathbf{z})$ were available, there are some risk allocations ϵ_i that result in higher performance than others, and predicting which of these allocations are better is nontrivial. This is explained as follows: since $\mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)$ can be thought of as a constant for given $\mathbf{F}_{\mathbf{z}}(\cdot)$ and ϵ, and since the variables ϵ_i are unspecified, there are infinite possible solutions for setting the ϵ_i variables (all solutions that lie on the boundary of $\sum_{i=1}^{N_a} \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i) \leq \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)$). Recall that the primary optimization objective of Eq. (6.1) is to maximize the chance-constrained scores by selecting plan values $\mathbf{x}$ and $\boldsymbol{\tau}$ that improve performance (i.e. $\max_{\mathbf{x},\boldsymbol{\tau}} y$), therefore a question that can be asked is: when setting agent risks according to Eq. (6.8), are some allocations of ϵ_i better than others, such that "better" plans $(\mathbf{x}, \boldsymbol{\tau})$ can be obtained? Answering this question is tricky, since obtaining "better" plans involves producing plans with "better" distributions $\mathbf{f}(\mathbf{z})$ and $\mathbf{f}(\mathbf{z}_i)$ such that the chance-constrained score $y = \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)$ is maximized. The complexity associated with this process is illustrated in Figure 6-2, which shows how the individual agent risks relate to the chance-constrained mission score given a distributed chance-constrained planning framework. As shown in the diagram, the agent risk allocations are used by the distributed planner to make agent plans, the score distributions associated with these agent plans are then convolved to derive the mission distribution, and finally the chance-constrained mission score can be computed given the mission score CDF and the allowable mission risk threshold. There are several parts of this process which make it virtually impossible to analytically derive the relationship between agent risks and chance-constrained mission score, but the most complex part is associated with the fact that the distributed planner will typically generate different plans given different risk allocations for ϵ_i. As a result, making predictions about chance-constrained mission performance given different agent risk allocations is very difficult and is thus an open research question.

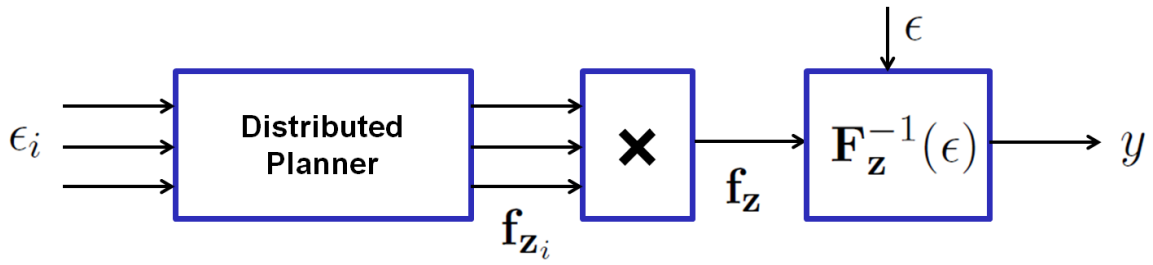


Figure 6-2: This figure shows the process relating individual agent risks to the chance-constrained mission score given a distributed chance-constrained planning framework. The main pieces include the distributed planner, which uses the risk allocations to make agent plans, a convolution block that combines the agent score distributions associated with the agent plans to derive the mission score distribution, and a final block that computes the chance-constrained mission score given the mission score distribution and the allowable mission risk threshold.

In this work, we develop heuristic strategies that approximate the planner output distributions for mission and agent scores, and use these to allocate risks amongst the agents within the CBBA framework according to Eq. (6.8). The next section describes how the distributed approximation to the chance-constrained problem presented here can be leveraged within the stochastic CBBA framework, and how the agent risks can be set using different heuristic strategies.

6.3 Chance-Constrained Extension to CBBA

6.3.1 Agent Risk Allocation Strategies

Although the decomposition in Equation (6.2) makes the problem easier to solve in a distributed fashion, it also introduces the additional complexity of picking the parameters ϵ_i such that the goal of maximizing the chance-constrained score of the mission distribution, $y = \sum_{i=1}^{N_a} y_i$, given the mission risk ϵ, is adequately represented. For generic task allocations, the relationship between the mission ϵ and each of the agents' ϵ_i's is nontrivial, as described in Section 6.2, however, given certain probability models, the complexity of picking these values can be reduced (note that doing this efficiently is still an open research question). This thesis addresses these issues by employing different heuristic strategies that attempt to model the individual agents' risks as a function of the global mission risk.

The heuristic risk allocation methods employed in this work use the expression provided by Eq. (6.8) to determine how to set the agent risks given the mission risk. The first case

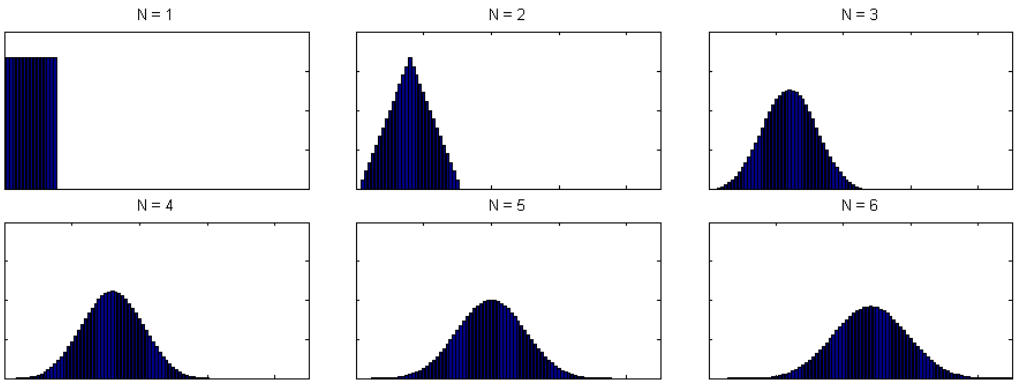


Figure 6-3: Illustration of the Central Limit Theorem, showing the distributions for sums of N uniform random variables for different values of N.

considered is for teams of homogeneous agents, where all agents in the team have similar planning parameters and underlying distributions. The heuristic strategies employed in this case assume that the distributions of the agent scores are all identical, and that the risk values ϵ_i will be the same for all agents. Using these assumptions, Eq. (6.8) reduces to

$$\epsilon_i = \mathbf{F}_{\mathbf{z}_i}\left(\frac{1}{N_a}\mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)\right) \tag{6.9}$$

where the agent risks ϵ_i are assumed to be identical. The expression in Eq. (6.9) can be used to describe any homogeneous team, however, specifying the mission distribution $\mathbf{f}(\mathbf{z})$ may be difficult given certain agent score distributions $\mathbf{f}(\mathbf{z}_i)$. In this work, we invoke the Central Limit Theorem, and use a Gaussian distribution to approximate the mission score (sum of agent score random variables), where the mean and variance of the distribution are given by the sum of the means and variances of the agent distributions respectively (i.e. $\mathbf{z} \sim \mathcal{N}\big(N_a\mu_i, N_a\sigma_i^2\big)$, where μ_i and σ_i^2 are the mean and variance of $\mathbf{f}(\mathbf{z}_i)$). As motivation for using this Gaussian assumption consider Figure 6-3, which shows the distributions for sums of N uniform random variables for different values of N. The baseline distribution is shown in the $N = 1$ frame, the distribution for a sum of two of these random variables is shown in the $N = 2$ frame, etc. As shown in these figures, the Central Limit Theorem converges very quickly, and with as few as 3 random variables the distributions of the sums look approximately Gaussian. Therefore, for multi-agent teams of 3 or more agents, approximating the mission scores as Gaussian is a reasonable assumption to make.

As a reminder, the CDF and inverse CDF expressions for a Gaussian distribution are

given by,

$$\begin{aligned} \mathbf{F}_X(x) &= \frac{1}{2}\left(1 + \text{ erf}\left(\frac{x-\mu}{\sqrt{2\sigma^2}}\right)\right) \\ \mathbf{F}_X^{-1}(\epsilon) &= \mu + \sqrt{2\sigma^2}\,\text{erf}^{-1}(2\epsilon - 1) \end{aligned} \tag{6.10}$$

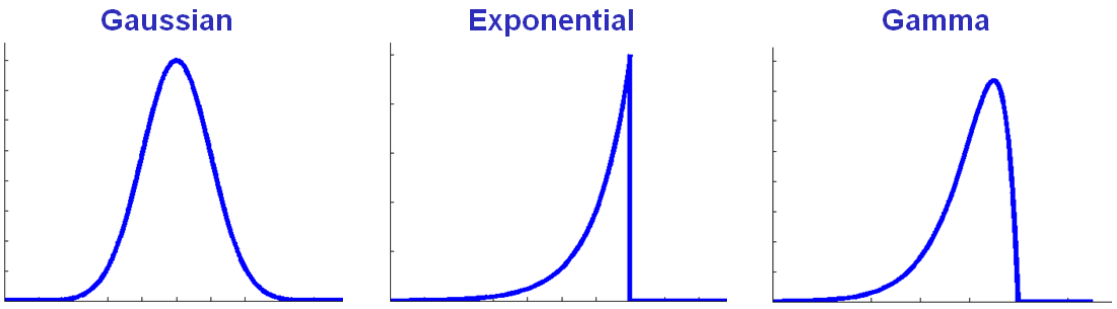
Using this Gaussian approximation for the mission score distribution, Eq. (6.9) can be written as

$$\epsilon_i = \mathbf{F}_{\mathbf{z}_i}\left(\mu_i + \sqrt{\frac{2}{N_a}}\sigma_i\,\text{erf}^{-1}(2\epsilon - 1)\right) \tag{6.11}$$

The expression provided in Eq. (6.11) can be used with many different forms of the agent distributions $\mathbf{f}(\mathbf{z}_i)$. The full derivation associated with these expressions is provided in Appendix A. In this thesis, we explored three different heuristics for homogeneous agents, assuming distributional forms based on Gaussian, exponential and gamma distributions for $\mathbf{f}(\mathbf{z}_i)$. These three heuristic strategies are derived in detail in Appendix A and are summarized in Eq. (6.12) below,

$$\begin{aligned} \textit{Gaussian} : \epsilon_i &= \frac{1}{2}\left(1 + \text{ erf}\left(\sqrt{\frac{1}{N_a}}\,\text{erf}^{-1}(2\epsilon - 1)\right)\right) \\ \textit{Exponential} : \epsilon_i &= e^{-\left(1-\sqrt{\frac{2}{N_a}}\,\text{erf}^{-1}(2\epsilon-1)\right)} \\ \textit{Gamma} : \epsilon_i &= 1 - \frac{1}{\Gamma(k)}\gamma\left(k, k - \sqrt{\frac{2k}{N_a}}\,\text{erf}^{-1}(2\epsilon - 1)\right) \end{aligned} \tag{6.12}$$

Illustrations of the agent score distribution forms used in these three different homogeneous risk heuristics are provided in Figure 6-4. The intuition behind using the two nonsymmetric distributions shown in the exponential and gamma cases was that, for the types of time-critical mission scenarios considered throughout this thesis, the score distributions for agents tended to have probability masses clustered around maximum task rewards and diminishing probabilities associated with obtaining lower scores. This was because arriving at a task on time or early resulted in agents receiving the full task score, whereas arriving late (but within the window of validity) resulted in exponentially decreasing task scores. The three heuristics presented in Eq. (6.12) were leveraged within the distributed CBBA framework, and used to plan for homogeneous teams operating in stochastic environments. Their performance is compared later in Section 6.4.



(a) Gaussian Risk (b) Exponential Risk (c) Gamma Risk

Figure 6-4: Agent score distributions used in the three different homogeneous risk heuristics.

Setting the risk values for heterogeneous agents is a bit more complicated, since the assumptions made in Eq. (6.9) regarding identical agent distributions and identical risk values may no longer hold. For general problems, Eq. (6.8) will have infinite possible combinations of ϵ_i as valid solutions for a given specific value of ϵ, therefore specifying different individual agent risks becomes difficult. There are two main goals associated with allocating risks amongst the agents. The first goal is that the risks given to individual agents should be such that the global mission risk level is adequately captured by the team. This was the purpose of Eq. (6.8) which identified a relationship between mission risk and agent risks given available plan distributions. The second goal is that the risks allocated to the agents should encourage agents to pick "better" plans, such that the chance-constrained mission score $\mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)$ be as high as possible. This involves finding a distribution for the mission score $\mathbf{z}$ that maximizes $\mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)$, however, $\mathbf{f}(\mathbf{z})$ is a function of the agent score distributions $\mathbf{f}(\mathbf{z}_i)$ (e.g. a convolution of these agent distributions if the agents are independent), and the distributions $\mathbf{f}(\mathbf{z}_i)$ are in turn functions of the risk levels ϵ_i and of the inner workings of the planner (which are hard to predict). This severe coupling makes the goal of optimizing the ϵ_i allotments to achieve the best plan very difficult. Another issue in distributed planning environments, is that the agents must be able to select their own values ϵ_i given statistics about the mission and the other agents, or must be able to share information with each other (e.g. distributions, moments, or even ϵ_i allocations) to converge on a consistent allocation of the risk levels ϵ_i.

With these issues in mind, this thesis considers a few different heuristic strategies to allocate risks amongst agents given a heterogeneous team. The different risk allocation strategies are explained in detain in Appendix A and are summarized in this section. The

first heuristic considered assumes that all agents are given identical risk values ϵ_i (note that this does *not* imply that the agents have identical distributions). Invoking the Central Limit Theorem again, the mission is assumed to be Gaussian, where the mission score distribution is given by $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$ with mean $\mu = \sum_{i=1}^{N_a} \mu_i$ and variance $\sigma^2 = \sum_{i=1}^{N_a} \sigma_i^2$, and Eq. (6.8) can be rewritten as,

$$\sum_{i=1}^{N_a} \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i) = \mu + \sqrt{2\sigma^2}\,\mathrm{erf}^{-1}(2\epsilon - 1) \tag{6.13}$$

Since the agent distributions are possibly all different, the left side of Eq. (6.13) is still difficult to compute, depending on the particular CDFs of the agent score distributions. In this work, we assume that agent distributions are also Gaussian, $\mathbf{z}_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, where the agent means and variances are assumed to be different since the team is heterogeneous. Using identical risk values and heterogeneous Gaussian distributions in Eq. (6.13), an analytic expression for the agent risks ϵ_i is given by,

$$\begin{aligned} \epsilon_i &= \frac{1}{2}\left(1 + \mathrm{erf}\left(\mathbf{H}\,\mathrm{erf}^{-1}(2\epsilon - 1)\right)\right) \\ \mathbf{H} &= \left(\frac{\sqrt{\sum_{i=1}^{N_a} \sigma_i^2}}{\sum_{i=1}^{N_a} \sqrt{\sigma_i^2}}\right) \end{aligned} \tag{6.14}$$

with the constant value $\mathbf{H}$ representing the team heterogeneity with regards to variance in agents' scores. This expression has several interesting properties. Firstly, the agent risk values for the Gaussian case do not depend on the means of the agent distributions or mission distribution, they only depend on the variances. This is similar to the observation made about the homogeneous risk allocation strategies, where the means of the distributions and scale parameters did not affect the risk allocation. However, in the heterogeneous case, the *relative* scale parameters do affect the risk allocation, as captured by the constant $\mathbf{H}$ in Eq. (6.14). Given the expression in Eq. (6.14), if the agents are homogeneous (with identical distributions), then $\mathbf{H} = 1/\sqrt{N_a}$ and the expression is equivalent to the homogeneous Gaussian risk allocation presented in Eq. (6.12). On the other hand, if the entire mission distribution comes from only 1 agent's contribution (all other agents are deterministic with no variance), then $\mathbf{H} = 1$ and $\epsilon_i = \epsilon$ as expected. This shows that team heterogeneity can be represented via the parameter $\mathbf{H}$, which is a function of N_a and of the relative

scales of the agent distributions with respect to one another. The range of $\mathbf{H}$ is given by $\mathbf{H} \in \left[\frac{1}{\sqrt{N_a}}, 1\right]$. A major advantage of using this heuristic versus more complex allocations between heterogeneous agents, is that agents can select a number within the range of $\mathbf{H}$ that roughly represents how heterogeneous the team is (possibly performing consensus on this number), and then use $\mathbf{H}$ to compute ϵ_i individually. This is significantly faster than coming to consensus on a consistent allocation of the individual parameters ϵ_i.

An alternate heuristic risk allocation strategy considered involves assigning different risk values ϵ_i to different types of agents, where agents of the same type would be assigned identical values of ϵ_i. For example, considering a scenario with 2 types of agents, and with equal numbers of each type of agent, Eq. (6.8) becomes,

$$\frac{N_a}{2}\mathbf{F}_{\mathbf{z}_1}^{-1}(\epsilon_1) + \frac{N_a}{2}\mathbf{F}_{\mathbf{z}_2}^{-1}(\epsilon_2) = \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon) \tag{6.15}$$

where the distributions and risks for each different agent type k are given by $\mathbf{z}_k$ and ϵ_k. Assuming Gaussian agent scores, and Gaussian mission scores as in the previous risk allocation strategy, Eq. (6.15) simplifies to the following expression,

$$\frac{\sigma_1}{\sqrt{\sigma_1^2+\sigma_2^2}}\,\mathrm{erf}^{-1}(2\epsilon_1 - 1) + \frac{\sigma_2}{\sqrt{\sigma_1^2+\sigma_2^2}}\,\mathrm{erf}^{-1}(2\epsilon_2 - 1) = \sqrt{\frac{2}{N_a}}\,\mathrm{erf}^{-1}(2\epsilon - 1) \tag{6.16}$$

Similar expressions can be derived given 3 or more types of agents. In Eq. (6.16), each of the terms on the left hand side include a scaling parameter that is proportional to the standard deviation for that agent type (normalized by the standard deviation of the mission). The right hand side of Eq. (6.16) includes the number of agents N_a and is typically a function of the number of agent types as well. Given the expression in Eq. (6.16), a key question involves deciding how to partition the risk amongst the agent types. This can be accomplished by splitting the right hand side of Eq. (6.16) into shares, and then solving for ϵ_k for each agent type k (where the agent risks would be set using $\epsilon_i = \epsilon_k$ for agents belonging to type k). It is not obvious, however, how these shares should be divided amongst the agent types. In this thesis we consider two cases, one involving equal shares, and one setting shares proportional to the standard deviation of the agent type σ_k.

The first strategy, which uses equal shares, divides the right hand side of Eq. (6.16) into

two equal parts, giving the following risk values ϵ_k for each group k,

$$\epsilon_k = \frac{1}{2}\left(1 + \text{ erf}\left(\left(\frac{\sqrt{\sigma_1^2 + \sigma_2^2}}{2\sigma_k}\sqrt{\frac{2}{N_a}}\right) \text{ erf}^{-1}(2\epsilon - 1)\right)\right) \tag{6.17}$$

The quantity preceding the inverse error function in Eq. (6.17) can be thought of as a scaling constant $\mathbf{H}_k$ to represent agent heterogeneity, where

$$\mathbf{H}_k = \left(\frac{\sqrt{\sigma_1^2 + \sigma_2^2}}{2\sigma_k}\sqrt{\frac{2}{N_a}}\right) \tag{6.18}$$

in Eq. (6.17). Different values of $\mathbf{H}_k$ will lead to different risk allocations ϵ_k, and again it is not immediately obvious how to partition the shares (how to set $\mathbf{H}_k$) to get an allocation of ϵ_i's for all agents that optimizes the chance-constrained mission score.

The second strategy used in this thesis assumes that the shares agents get are proportional to their standard deviation, thus the right hand side of Eq. (6.16) is divided into shares of size $\sigma_k / \sum_k \sigma_k$. Given this division, each group computes the following risk thresholds ϵ_k given by,

$$\epsilon_k = \frac{1}{2}\left(1 + \text{ erf}\left(\left(\frac{\sqrt{\sigma_1^2 + \sigma_2^2}}{\sigma_1 + \sigma_2}\sqrt{\frac{2}{N_a}}\right) \text{ erf}^{-1}(2\epsilon - 1)\right)\right) \tag{6.19}$$

where the constant $\mathbf{H}_k$ becomes

$$\mathbf{H}_k = \left(\frac{\sqrt{\sigma_1^2 + \sigma_2^2}}{\sigma_1 + \sigma_2}\sqrt{\frac{2}{N_a}}\right) \tag{6.20}$$

Note that in this case, the constant $\mathbf{H}_k$ is not explicitly dependent on the individual parameter σ_k anymore, but rather considers statistics over the variances for all agent types. As a result, $\mathbf{H}_k$ will be constant for all agent types k and therefore the risk values ϵ_k will all be the same, leading to equal risks for all agents in the team (but still capturing the heterogeneity associated with the different variances for agent scores). It is shown in Section 6.4 that this last strategy, where risks are equal for all agents, performs significantly better than the strategy shown in Eq. (6.17). This is because by balancing the risks more evenly throughout the team, no agent can take on its extreme plan values (very deterministic taking no risk at all, or taking too much risk and not considering how it might affect the mission as a whole), which increases the performance of the team. Furthermore, the

heuristic strategy employed in Eq. (6.14), which also assigned equal risks to all the agents, also achieved high performance which was significantly better than the strategy shown in Eq. (6.17), and on par with the strategy of Eq. (6.19). On closer inspection, the value for $\mathbf{H}_k$ in Eq. (6.19) looks very similar to that of $\mathbf{H}$ in Eq. (6.14), explaining why the performance of both heuristics was similar, since they both capture the same relative scaling effects associated with the heterogeneous agent variances.

The different heuristics described in this section can be used within the CBBA framework to allocate risk amongst the agents given knowledge of the team heterogeneity and approximations of the agent score distribution forms. The performance of these different heuristics is compared later in Section 6.4. Of course, there are many other heuristics that can be used, and designing risk allocation strategies that work well for general scenarios of interest remains an active area of research. The next section describes how the distributed chance-constrained approximation can be utilized within the CBBA framework given a specific risk allocation for the agents.

6.3.2 Stochastic Bundle Construction

The previous section specified how to allocate risk amongst the agents using different heuristic strategies. Given these individual risk allotments ϵ_i, each agent can solve its own distributed chance-constrained optimization, as specified in Eq. (6.2), to select its best set of assignments. This section describes how the robust CBBA framework proposed in Chapter 5 can be used to solve the distributed chance-constrained approximation. Within the bundle construction phase of CBBA, the greedy process by which each agent constructs its task bundle needs to ensure that, for every bid, the task scores satisfy the probabilistic constraints specified by Eq. (6.2). This problem involves solving the following optimization for all available tasks $j \in \mathcal{J} \setminus \mathbf{p}_i$, where each task j is inserted into the path at all possible path locations n_j,

$$
\begin{aligned}
J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} = \max_{n_j} \quad & y_i \\
\text{s.t.} \quad & \mathbb{P}_{\boldsymbol{\theta}} \left\{ \left(\sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^\star(\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta}) \; x_{ij} \right) > y_i \right\} \geq 1 - \epsilon_i
\end{aligned} \tag{6.21}
$$

Here the chance-constraint captures the effect of the uncertainty on the entire bundle given the probabilistic constraint. As described previously in Chapter 5, the uncertainty in the system affects the times at which tasks will be executed, thus affecting their scores (e.g. inserting a task into the path will impact arrival times for all subsequent tasks, subject to the uncertainty in the system). The chance-constrained calculation in Eq. (6.21) accounts for this coupling between task scores. The marginal score for each task j is then given by the increase in y_i as a result of adding task j,

$$\Delta J_{ij}(\mathbf{p}_i) = J_{(\mathbf{p}_i \oplus_{n_j^\star} j)} - J_{\mathbf{p}_i}$$

and the optimal task to add is given by,

$$j^\star = \underset{j \notin \mathbf{p}_i}{\text{argmax}} \ \Delta J_{ij}(\mathbf{p}_i)\ h_{ij} \tag{6.22}$$

where h_{ij} is again computed using the warped bid, $s_{ij} = \min(\Delta J_{ij}(\mathbf{p}_i), y_{ik}),\ \forall k \in \mathbf{p}_i$, as described in Section 5.2.2. The bundle, path, times, winning agents list, and winning bids list are then updated to include the new task,

$$\begin{aligned}
\mathbf{b}_i &\leftarrow (\mathbf{b}_i \oplus_{\text{end}} j^\star) \\
\mathbf{p}_i &\leftarrow (\mathbf{p}_i \oplus_{n_j^\star} j^\star) \\
\boldsymbol{\tau}_i &\leftarrow (\boldsymbol{\tau}_i \oplus_{n_j^\star} \tau^\star_{ij^\star}(\mathbf{p}_i \oplus_{n_j^\star} j^\star)) \\
z_{ij^\star} &= i \\
y_{ij^\star} &= s_{ij}
\end{aligned}$$

Two complications arise with this process, similar to the issues described previously in Chapter 5:

1. The first is that the task execution times are random variables that are subject to the uncertainty in the system, which makes the step of computing the "optimal" execution times nontrivial, since these times may be different for different realizations of the uncertainty. Therefore, when optimizing the path score in Eq. (6.21), the computation must take into account that different optimal execution times may result for the same path given different values of $\boldsymbol{\theta}$. Again, analytically computing these time-

optimization effects is usually intractable, motivating the use of numerical methods to approximate the score calculation in Eq. (6.21).

2. The second issue is that computing the probabilities of the score function specified in Eq. (6.21) usually involves finding the distribution of a sum of non-independent heterogeneous task scores. This expression is again analytically intractable for most types of problems due to nontrivial coupling of distributions within the score function, motivating the use of sampling methods.

The numerical approach employed to sample the score function in Eq. (6.21) addresses these tractability issues and is described later in this section (see Algorithm 9). The use of marginal scores within the bundle construction process allows the algorithm to appropriately represent the impact of the probabilistic chance-constraint during every iteration. Therefore, even though the bundle is being constructed sequentially, computing the marginal score for tasks requires computing the effect of adding each task on the entire bundle in the associated probabilistic constraint, therefore the coupling between tasks is correctly captured within a consistent framework.

To approximate the probabilistic score functions, this work employed a sampling approach to generate a set of representative samples, $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$, with associated weights, $\{w_1, \ldots, w_N\}$, that approximate the distribution of $\boldsymbol{\theta}$. Using sampling, an approximation to the score function used in the bundle construction process involves executing the following steps, which are summarized in Algorithm 9:

1. For each sample value $\boldsymbol{\theta}_k$, the score for task j being inserted at location n_j can be deterministically computed by,

$$J_{(\mathbf{p}_i \oplus_{n_j} j)k} = \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}^{\star}(\mathbf{p}_i \oplus_{n_j} j), \boldsymbol{\theta}_k)\ x_{ij}$$

   Note that this step involves optimizing the task execution times $\tau_{ij}^{\star}$ given the specific realization of the uncertain random variables $\boldsymbol{\theta}_k$. Even though this process involves optimizing the set of task execution times in continuous time, for the decaying cost functions described previously in this thesis (see Section 5.3), there are only a discrete number of continuous times that could maximize this sample's score for the task, therefore computing the optimal task times remains tractable.

2. Step 1 is repeated for all N samples, $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$, where N must be large enough to create a representative probability distribution of the scores obtained from adding this task.

3. The samples are then sorted in increasing order based on the values of the sampled scores, $J_{(\mathbf{p}_i \oplus_{n_j} j)k}$. We define the variable that indexes this ordered set as $\bar{k} \in \{1, \ldots, N\}$. The resulting sorted sampled scores, along with their probabilistic weights w_k, describe a discrete probability distribution that approximates the true score PDF. As a reminder, finding the value y_i in the chance-constrained formulation given risk threshold ϵ_i is equivalent to integrating the PDF up to ϵ_i and determining what value of y_i this corresponds to. Given the discrete PMF specified through the sampling process, this integration can be approximated by summing the weights of the ordered samples until the threshold ϵ_i is reached. This can be written formally as,

$$\max_{\bar{k} \in \{1,\ldots,N\}} \left\{ J_{(\mathbf{p}_i \oplus_{n_j} j)\bar{k}} \;\middle|\; \sum_{i=1}^{\bar{k}} w_i \leq \epsilon_i \right\} \tag{6.23}$$

   where the resulting $J_{(\mathbf{p}_i \oplus_{n_j} j)\bar{k}^\star}$ provides a lower bound on the path score within the allowable risk threshold. The full stochastic bundle construction process is equivalent to the one described previously in Algorithm 8, where the COMPUTE-STOCHASTIC-PATH-SCORE($\mathbf{p}_i \oplus_{n_j} j$) function is replaced with the chance-constrained score calculation described in this section (see Algorithm 9).

As mentioned in the previous chapter, another advantage of using sampling is that, although stochastic planning increases the computational complexity of the planning process with respect to the deterministic formulation, the number of samples can be adjusted given the available computational resources. Therefore, the chance-constrained extension to CBBA preserves polynomial-time convergence (although the plan time increases linearly with the number of samples N).

6.4 Example Applications

The distributed chance-constrained CBBA algorithm was implemented in simulation and tested on time-varying UAV missions similar to those described in Section 5.3. The first set of experiments involved a stochastic mission with 6 homogeneous agents and 60 tasks, where

Algorithm 9 COMPUTE-STOCHASTIC-PATH-SCORE($\mathbf{p}_i$) - (Chance-Constrained)

1: $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\} \sim \mathbf{f}(\boldsymbol{\theta})$

2: $\{w_1, \ldots, w_N\} \leftarrow \{w_1, \ldots, w_N\} / \sum_{k=1}^{N} w_k$

3: **for** $k \in \{1, \ldots, N\}$ **do**

4: $\quad \boldsymbol{\tau}_i^\star = \underset{\boldsymbol{\tau}_i}{\text{argmax}} \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau_{ij}(\mathbf{p}_i), \boldsymbol{\theta}_k)\ x_{ij}$

5: $\quad J^k_{\mathbf{p}_i} = \sum_{j=1}^{N_t} \mathbf{c}_{ij}(\tau^\star_{ij}(\mathbf{p}_i), \boldsymbol{\theta}_k)\ x_{ij}$

6: **end for**

7: $\left(J^{\bar{k}}_{\mathbf{p}_i}, w_{\bar{k}}\right) \leftarrow \text{SORT-SCORES}\left(J^k_{\mathbf{p}_i}, w_k\right)$

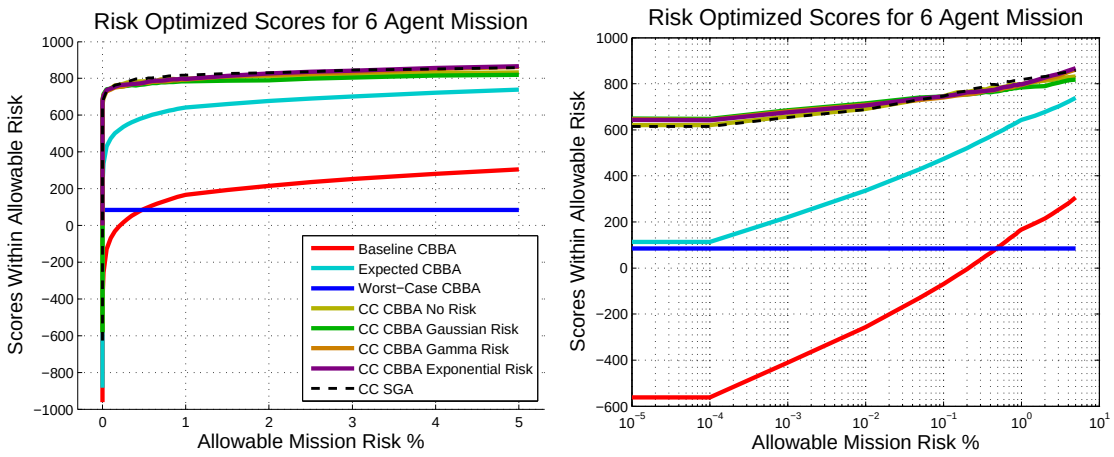
8: $\bar{k}^\star = \underset{\bar{k} \in \{1,\ldots,N\}}{\text{argmax}} \left\{ J^{\bar{k}}_{\mathbf{p}_i} \;\middle|\; \sum_{i=1}^{\bar{k}} w_i \leq \epsilon_i \right\}$

9: **return** $(J^{\bar{k}^\star}_{\mathbf{p}_i})$

the team had to optimize performance for various mission risk thresholds. In this scenario, task durations were uncertain and were distributed according to a gamma distribution. There were three types of tasks: high-reward high-uncertainty tasks, medium-reward tasks with low variance, and deterministic tasks with much lower rewards. The UAV team was also composed of homogeneous agents with uncertain velocities (with a uniform distribution). In the second set of experiments, the task scenario was equivalent, but the team was now composed of heterogeneous agents consisting of fast but unpredictable agents (high mean and high variance), and slower speed but more predictable agents (low mean and low variance), both having uniform distributions on velocities. The following sections discuss the results for these two different scenarios.

6.4.1 Homogeneous Agents

Figure 6-5 show Monte Carlo simulation results for a stochastic mission with 6 homogeneous agents showing chance-constrained mission performance as a function of the mission risk level for different planning algorithms. In the experiments, the following 8 planning algorithms were compared: the Baseline (deterministic) CBBA algorithm, the Expected-Value CBBA algorithm from Chapter 5, the Worst-Case CBBA algorithm from Chapter 5, the Chance-Constrained CBBA algorithm proposed in this chapter, but with no risk allocation between the agents (all agents planned with the mission risk $\epsilon_i = \epsilon$ which typically leads



(a) Chance-constrained scores

(b) Chance-constrained scores (log scale)

Figure 6-5: Monte Carlo simulation results for a stochastic mission with 6 homogeneous agents showing chance-constrained mission performance as a function of the mission risk level. The plots show the chance-constrained mission scores (worst score within allowable risk threshold) for 8 different planning algorithms: Baseline (deterministic) CBBA, the Expected-Value CBBA algorithm, the Worst-Case CBBA algorithm, the proposed Chance-Constrained CBBA algorithm with no risk allocation, Chance-Constrained CBBA using Gaussian risk allocation, Chance-Constrained CBBA using Gamma risk allocation, Chance-Constrained CBBA using Exponential risk allocation, and a centralized chance-constrained sequential greedy algorithm (SGA). Figure (a) shows the chance-constrained mission scores as a function of mission risk, and Figure (b) shows the same information as Figure (a) but on a log-scale to highlight the performance at low risk levels.

to a very conservative estimate of the risk), the Chance-Constrained CBBA using the different risk allocation strategies including Gaussian, exponential and gamma, and finally a centralized chance-constrained sequential greedy algorithm (SGA) planning for all agents simultaneously. Figure 6-5 shows the chance-constrained mission scores (worst score within allowable risk threshold) for the 8 different planning algorithms, where Figure 6-5(a) shows the chance-constrained mission scores as a function of mission risk, and Figure 6-5(b) shows the same scores on log-scale to highlight the performance at low risk levels. As seen in the plots, all the chance-constrained planning approaches do significantly better than the baseline (deterministic) CBBA algorithm and the 2 robust CBBA variants presented in Chapter 5.

To highlight the difference between these chance-constrained approached, Figure 6-6 shows a close up of the mission scores achieved plotted on a linear scale (Fig. 6-6(a)) and on a log scale (Fig. 6-6(b)). The results show that the three heuristics algorithms achieved

higher performance than without a risk allocation strategy, and the the exponential heuristic performed best at higher risk thresholds, whereas the Gaussian risk performed better at lower risk thresholds (although the performance of all three strategies was very close, especially at low risk thresholds). Figures 6-6(b) and 6-6(d) show the achieved mission risk level corresponding to the individual agent risk levels ϵ_i. The dotted line on the plots represents a perfect match between desired and actual mission risk. As shown in the figures, without risk allocation the team performs conservatively, achieving a much lower mission risk than allowed and thus sacrificing performance. With the risk allocation methods, the team is able to more accurately predict the mission risk, achieving higher performing plans within the allowable threshold. The results show that when algorithms match the mission risk well the performance of the planner increases, further motivating the need for good risk allocation strategies. Furthermore, the distributed Chance-Constrained CBBA planner achieves performance on par with the centralized sequential greedy approach, thus validating the distributed approximation to the centralized chance-constrained problem. Finally Figure 6-7 shows histograms for the mission scores (Fig. 6-7(a)) comparing the baseline (deterministic) CBBA, the worst-case robust CBBA, and Chance-Constrained CBBA using the exponential risk allocation heuristic. As seen in the plots, the distributions for the multi-agent mission scores are nearly Gaussian, justifying the use of the Gaussian approximation for mission scores in the risk allocation heuristics. Figure 6-7(b) shows the individual agent distributions, showing how the exponential and gamma distribution approximations (explained fully in Appendix A) are good choices to approximate the agent scores distributions.

6.4.2 Heterogeneous Agents

The second set of experiments involved a heterogeneous team with different statistical properties on their uncertain planning parameters. Figure 6-8 show Monte Carlo simulation results for a stochastic mission with 6 heterogeneous agents showing chance-constrained mission performance as a function of the mission risk level for different planning algorithms. In the experiments, the following 8 planning algorithms were compared: the Baseline (deterministic) CBBA algorithm, the Expected-Value CBBA algorithm from Chapter 5, the Worst-Case CBBA algorithm from Chapter 5, the Chance-Constrained CBBA algorithm proposed in this chapter, but with no risk allocation between the agents (all agents planned with the mission risk $\epsilon_i = \epsilon$), the Chance-Constrained CBBA algorithm using the 3 different

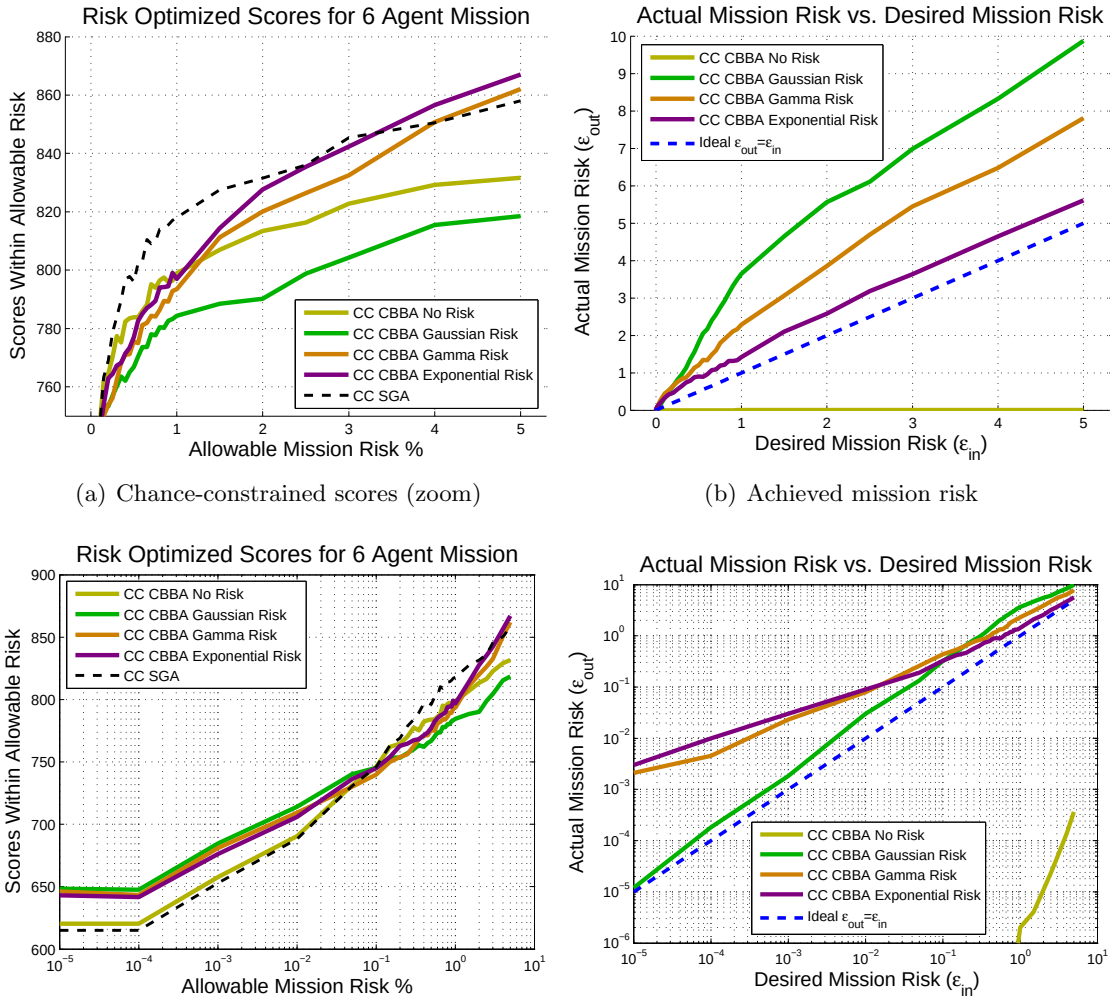
heterogeneous risk allocation strategies described in Section 6.3.1, and finally a centralized chance-constrained sequential greedy algorithm (SGA) planning for all agents simultaneously. The 3 risk allocation strategies consisted of the equal risk heuristic approximation described in Eq. 6.14 (termed "Risk 1"), the heterogeneous risk allocation using equal shares described in Eq. 6.17 (termed "Risk 2"), and the heterogeneous risk allocation strategy with risks proportional to agents standard deviations ("Risk 3", which effectively led to equal risks amongst agents). Figure 6-8 shows the chance-constrained mission scores (worst score within allowable risk threshold) for the 8 different planning algorithms, where Figure 6-8(a) shows the chance-constrained mission scores as a function of mission risk, and Figure 6-8(b) shows the same scores on log-scale to highlight the performance at low risk levels. As seen in the plots, all the chance-constrained planning approaches do significantly better than the baseline (deterministic) CBBA algorithm and the 2 robust CBBA variants presented in Chapter 5.

To highlight the difference between these chance-constrained approached, Figure 6-9 shows a close up of the mission scores achieved plotted on a linear scale (Fig. 6-9(a)) and on a log scale (Fig. 6-9(b)). The results show that the two heuristics algorithms with equal risks achieved higher performance than without a risk allocation strategy, however, the heterogeneous risk allocation strategy with different agent risks performed rather poorly. Figures 6-9(b) and 6-9(d) show the achieved mission risk level corresponding to the individual agent risk levels ϵ_i. The dotted line on the plots represents a perfect match between desired and actual mission risk. As shown in the figures, without risk allocation the team performs conservatively, achieving a much lower mission risk than allowed and thus sacrificing performance. With the equal risk allocation methods, the team is able to more accurately predict the mission risk, achieving higher performing plans within the allowable threshold. The results show that when algorithms match the mission risk well the performance of the planner increases, in general, however, as seen in the results for "Risk 2", even though the achieved risk levels were similar to "Risk 3" the performance was significantly lower. This is due to the fact that agents had unequal distributions, therefore some agents developed really aggressive plans whereas others selected plans that were too conservative. In general, having a more equitable risk distribution for the team led to higher performing plans. In general, however, it is hard to determine these effects in advance further motivating the need for good risk allocation strategies. Once again, the distributed Chance-Constrained CBBA

planner achieved performance on par with the centralized sequential greedy approach, thus validating the distributed approximation to the centralized chance-constrained problem.

Finally Figure 6-10 shows histograms for the mission scores (on the left) comparing the baseline (deterministic) CBBA, the worst-case robust CBBA, and Chance-Constrained CBBA using the different risk allocation strategies. As seen in the plots, the distributions for the multi-agent mission scores are nearly Gaussian, justifying the use of the Gaussian approximation for mission scores in the risk allocation heuristics. The left set of plots in Figure 6-10 show the individual agent distributions for the different chance-constrained algorithms. The first and third risk strategies (with equal risks for all agents) achieved similar looking agent distributions which were well balanced between the agents. The second risk strategy, where agents planned with heterogeneous risk allotments, achieved different score performance between the predictable agents (right) and the unpredictable agents (left). In particular, agents assigned a higher risk had distributions that were too spread out whereas agents planning with very low risk had distributions that were too conservative (very low variance). As a result, the chance-constrained score associated with the overall team distribution was lower (Fig. 6-10(c))

This chapter presented a distributed chance-constrained task allocation framework that can be used to plan for multi-agent networked teams operating in stochastic and dynamic environments. The results in both this chapter and Chapter 5 showed that by explicitly accounting for uncertainty propagation during the task allocation process large improvements can be made for distributed multi-agent teams operating in stochastic environments.



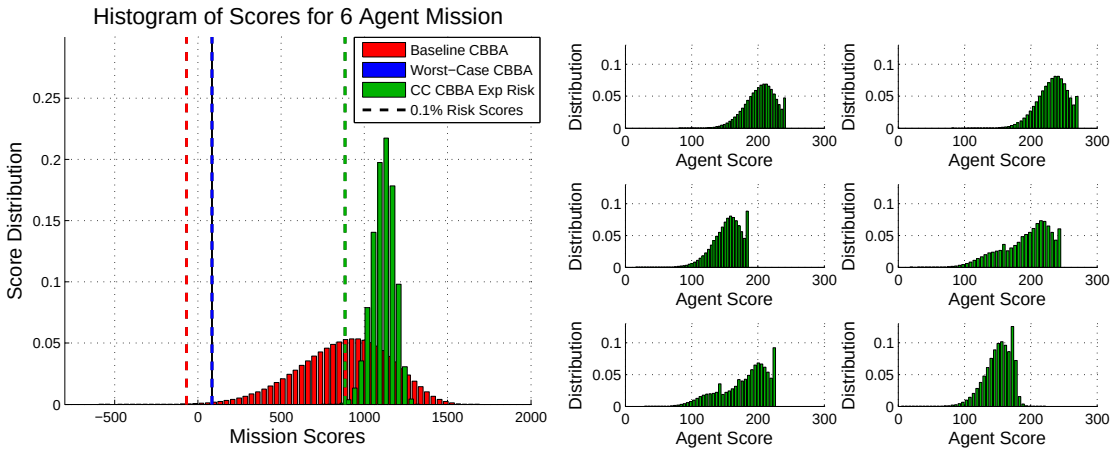
(a) Chance-constrained scores (zoom)

(b) Achieved mission risk

(c) Chance-constrained scores (zoom, log scale)

(d) Achieved mission risk (log scale)

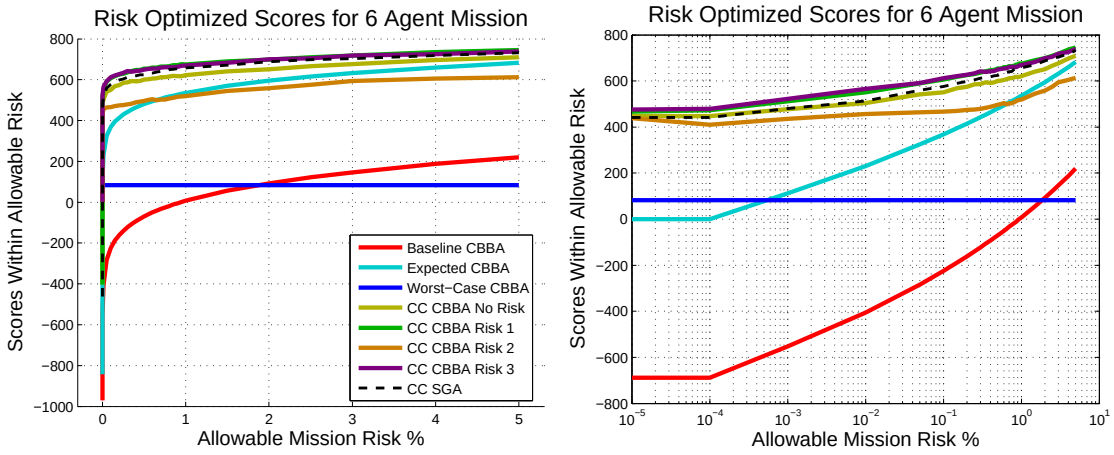
Figure 6-6: Monte Carlo simulation results for a stochastic mission with 6 homogeneous agents, comparing the performance of Chance-Constrained CBBA using different risk allocation strategies. Figure (a) shows a zoomed-in version of Figure 6-5(a); (b) shows the achieved mission risk corresponding to the distributed risk approximation for the different risk allocation strategies, versus desired mission risk; (c) shows a zoomed-in version of Figure 6-5(b) on a log scale to highlight the performance at low risk levels; and (d) shows the achieved mission risk on a log scale highlighting the performance at low risk levels.



(a) Histogram of mission scores

(b) Histograms of agent scores

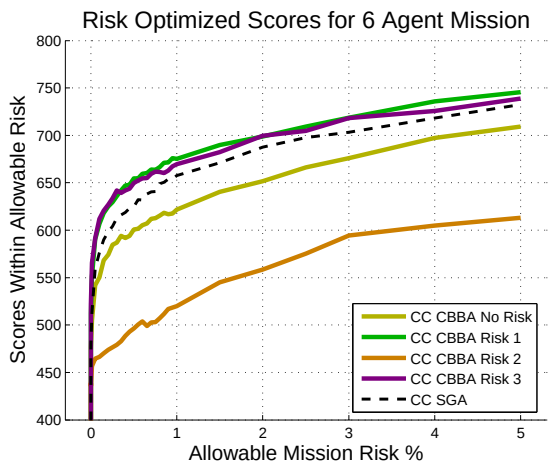
Figure 6-7: Simulation results for a stochastic mission with 6 homogeneous agents showing the achieved distributions and chance-constrained mission scores. Figure (a) shows histograms and chance-constrained scores for Baseline (deterministic) CBBA, Worst-Case CBBA, and Chance-Constrained CBBA using a 0.1% risk level. Figure (b) shows histograms of the scores achieved by the 6 agents using the Chance-Constrained CBBA algorithm with exponential risk allocation.



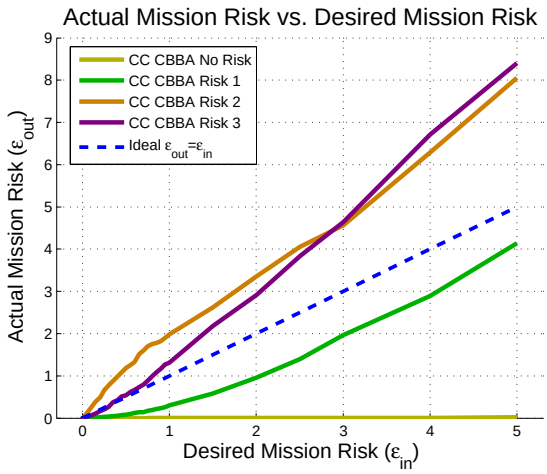
(a) Chance-constrained scores

(b) Chance-constrained scores (log scale)

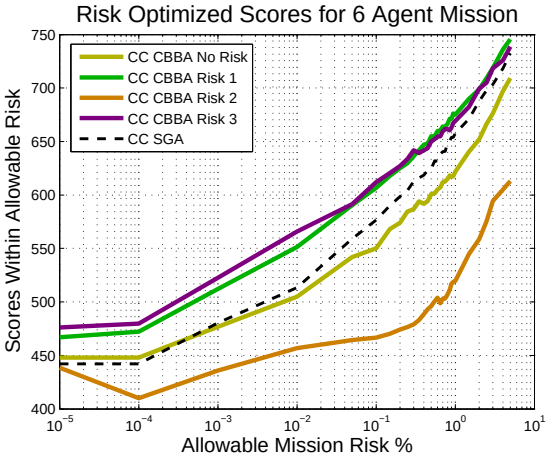
Figure 6-8: Monte Carlo simulation results for a stochastic mission with 6 heterogeneous agents showing chance-constrained mission performance as a function of the mission risk level. The plots show the chance-constrained mission scores (worst score within allowable risk threshold) for 8 different planning algorithms: Baseline (deterministic) CBBA, the Expected-Value CBBA algorithm, the Worst-Case CBBA algorithm, the proposed Chance-Constrained CBBA algorithm with no risk allocation, Chance-Constrained CBBA using 3 different risk allocation strategies (described in the text), and finally a centralized chance-constrained sequential greedy algorithm (SGA). Figure (a) shows the chance-constrained mission scores as a function of mission risk, and Figure (b) shows the same information as Figure (a) but on a log-scale to highlight the performance at low risk levels.



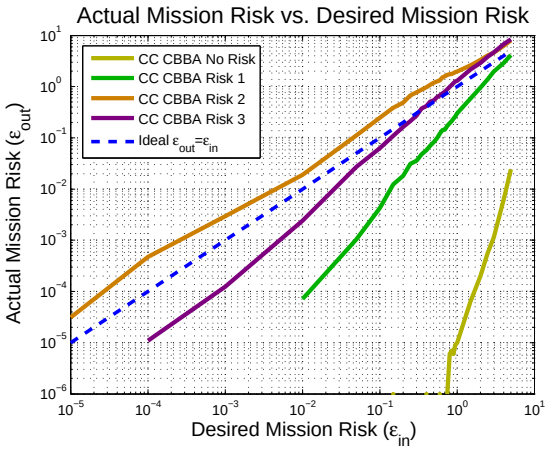
(a) Chance-constrained scores (zoom)



(b) Achieved mission risk

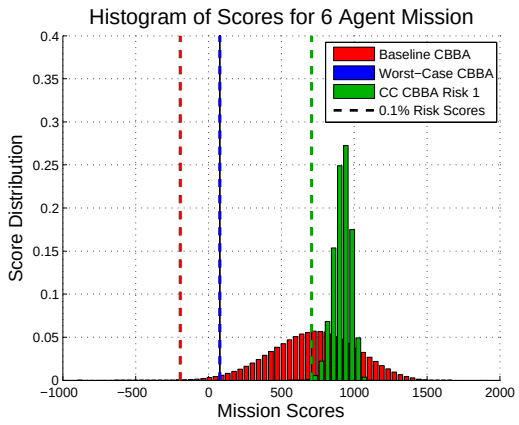


(c) Chance-constrained scores (zoom, log scale)

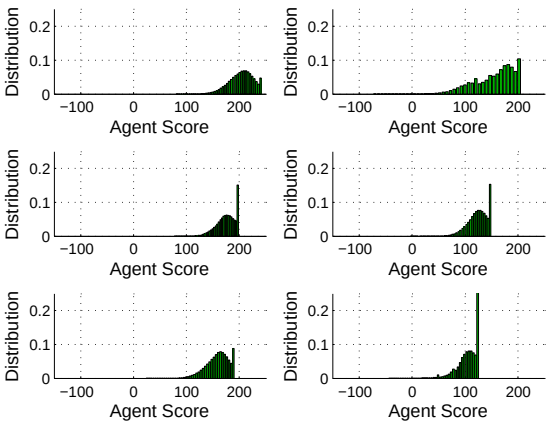


(d) Achieved mission risk (log scale)

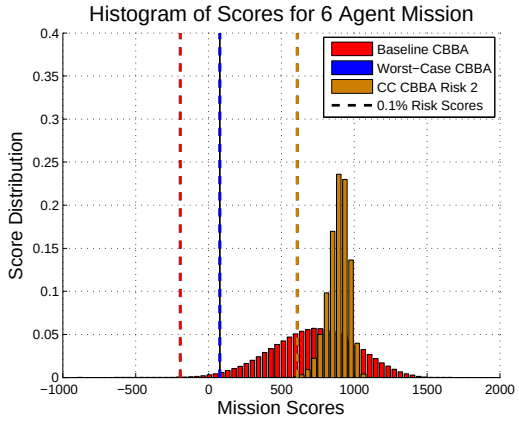
Figure 6-9: Monte Carlo simulation results for a stochastic mission with 6 heterogeneous agents, comparing the performance of Chance-Constrained CBBA using different risk allocation strategies. Figure (a) shows a zoomed-in version of Figure 6-8(a); (b) shows the achieved mission risk corresponding to the distributed risk approximation for the different risk allocation strategies, versus desired mission risk; (c) shows a zoomed-in version of Figure 6-5(b) on a log scale to highlight the performance at low risk levels; and (d) shows the achieved mission risk on a log scale highlighting the performance at low risk levels.



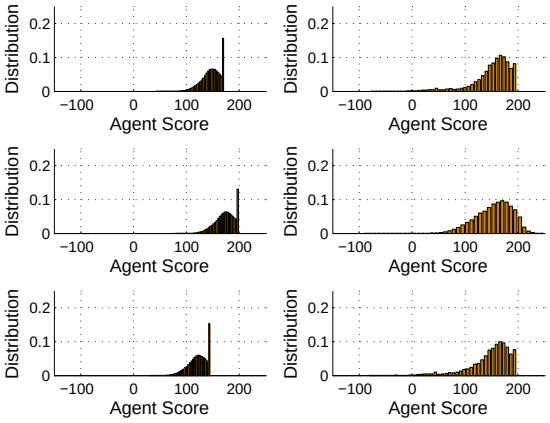
(a) Histogram of mission scores (Risk 1)



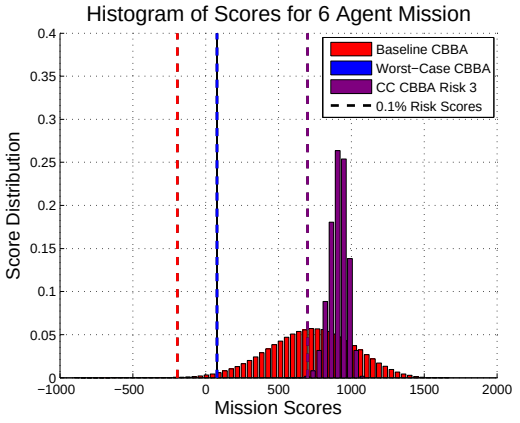
(b) Histograms of agent scores (Risk 1)



(c) Histogram of mission scores (Risk 2)



(d) Histograms of agent scores (Risk 2)



(e) Histogram of mission scores (Risk 3)

(f) Histograms of agent scores (Risk 3)

Figure 6-10: Simulation results for a stochastic mission with 6 heterogeneous agents. The figures on the left show achieved mission distributions and chance-constrained scores for Baseline CBBA, Worst-Case CBBA, and Chance-Constrained CBBA using different risk allocation strategies, for a 0.1% mission risk level. The figures on the right show histograms of the individual agent scores achieved using Chance-Constrained CBBA with the different risk allocation strategies.

Chapter 7

Conclusions

7.1 Summary of Contributions

This thesis addressed the problem of real-time robust distributed planning for multi-agent networked teams operating in uncertain and dynamic environments. In particular, several extensions and variants to the baseline CBBA algorithm presented in [58] were proposed and discussed, enabling distributed real-time planning in time-critical, communication-limited, and uncertain environments. The specific contributions of this thesis are described as follows:

1. This thesis extended CBBA to handle time-critical mission considerations, where time-varying score functions could be optimized within the CBBA algorithmic framework to enable dynamic planning for agents and tasks with specific timing constraints (e.g. task time-windows of validity, time-varying rewards for time-critical tasks, agent velocities). In particular, the CBBA with Time-Varying Score Functions algorithm proposed in Section 4.2 modified the bundle construction process of CBBA to explicitly optimize task execution times as well as agent assignments, enabling both spatial and temporal coordination of multi-agent teams in dynamic mission scenarios. The algorithm performance was validated through simulations, and a real-time replanning architecture was designed and implemented to enable real-time experiments for heterogeneous networked teams. Flight test experiments involving multi-agent dynamic search and track missions were performed in an indoor flight test facility at the MIT Aerospace Controls Lab using heterogeneous teams of quadrotor UAVs and robotic ground vehicles, demonstrating the real-time applicability of the distributed planning

algorithms.

2. This thesis extended the CBBA planning framework to enable conflict-free distributed planning in the presence of network disconnects due to communication-limited operating environments. The proposed approach, described in Section 4.3, employed a local distributed task space partitioning strategy, where the sets of tasks available to the different agent sub-networks were disjoint, thus ensuring conflict-free solutions. Simulation and experimental flight tests validated the proposed algorithms, showing improved performance over the baseline CBBA algorithm, but with lower communication and computational overhead than centralized strategies which require *a priori* task space partitioning at every replan iteration.

3. In Section 4.4, we developed a distributed cooperative planning algorithm that built upon the CBBA framework to enable agents to maintain network connectivity in communication-limited environments. The algorithm, named CBBA with Relays, guarantees network connectivity for agents performing tasks that require a live connection (e.g. video streaming), by locally incentivizing under-utilized agents to act as communication relays for other agents within a distributed framework. The proposed algorithm explicitly handles the joint network connectivity constraints through a local distributed network prediction phase, and cooperation between agents is enabled through the local creation of relay tasks by the agents that require a network connection. The CBBA with Relays algorithm is guaranteed to converge, runs in real-time, and guarantees network connectivity while tasks are being executed. The algorithm was validated through simulation, and indoor and outdoor flight test experiments, demonstrating real-time applicability.

4. In Chapter 5, we extended the CBBA with Time-Varying Score Functions algorithm of Section 4.2 to explicitly account for robustness in the planning process. A real-time distributed robust planning framework, named Robust CBBA, was proposed, which can leverage probabilistic models of planning parameters and different distributable stochastic metrics to hedge against parameter uncertainty. The algorithm employed sampling approaches to compute agent path scores given different stochastic metrics within the CBBA bundle construction process, enabling polynomial-time algorithm convergence. The Robust CBBA framework leverages a recent submodular extension

of CBBA proposed by Johnson [106] to guarantee distributed algorithm convergence given different stochastic metrics, and uses the convergence guarantees of CBBA under varying situational awareness to allow agents to individually construct their robust plans given local uncertainty representations. The Robust CBBA algorithm was implemented using two stochastic metrics, the expected-value metric and the worst-case stochastic metric, and used to plan for heterogeneous multi-agent teams performing search and track missions in uncertain environments. Simulation results were provided demonstrating real-time applicability, and showing that Robust CBBA improves performance over the baseline CBBA algorithm and achieves results similar to centralized planning strategies, validating the distributed approach.

5. In Chapter 6, we extended the Robust CBBA framework proposed in Chapter 5 to optimize performance in environments where low probability of failure is required. The approach used a chance-constrained stochastic metric that provides probabilistic guarantees on achievable mission performance given allowable risk thresholds. A distributed approximation to the chance-constrained metric was proposed to enable the use of Robust CBBA in these risk-aware environments, and constraints on individual risk allocations were derived to guarantee equivalence between the centralized chance-constrained optimization and the distributed approximation. Different risk allocation strategies for homogeneous and heterogeneous teams were proposed that approximate the agent and mission score distributions *a priori*, and results were provided comparing the performance of these in time-critical mission scenarios. The distributed chance-constrained CBBA algorithm was validated through simulation trials, and the results showed improved performance over baseline CBBA and over worst-case conservative planning strategies given allowable risk thresholds. Furthermore, the distributed chance-constrained approximation algorithm proposed in Chapter 6 achieved similar results to those obtained by centralized chance-constrained methods, validating the distributed approximation.

7.2 Future Work

There are several areas in which this research could be further extended. This section highlights a few promising research directions and discusses the associated challenges and

benefits of each.

In realistic mission scenarios, agent score functions are often coupled through environmental effects that impact all agents simultaneously (e.g. wind, magnetic variations, poor visibility, etc.), however, as mentioned in Section 5.1.1, handling coupling between agent score functions within a distributed planning framework is a nontrivial endeavor, and is thus an open area of research. In these types of scenarios, the distributed Robust CBBA framework presented in this thesis can still be used and is still guaranteed to converge, however, the performance of the team may suffer if the coupling is not explicitly considered. To improve team performance in these coupled stochastic environments, agents would need to share information about the uncertainty to capture the coupling explicitly throughout the planning process. Sharing samples between agents would typically require too much communication and thus seems impractical. Current research is considering hyper-parameter consensus methods [83], where agents can share knowledge about planning parameter distributions (means, variances, etc.) to improve situational awareness and thus increase team performance. In scenarios with explicit coupling, agents might want to share a few representative sample values (e.g. worst-case samples, sigma-points, samples of largest KL-divergence from neighboring agents' distributions, etc.), but deciding what these should be is a nontrivial question motivating further research.

Within the Robust CBBA framework, sampling algorithms were employed to approximate the different stochastic metrics while maintaining analytic and computational tractability. For stochastic metrics focusing on low probability events (e.g. chance-constrained or worst-case metrics), several rare-event or importance sampling methods have been successfully used throughout the literature to reduce the number of samples required by the algorithms [11]. Implementing these within the Robust CBBA framework is nontrivial given the complex distribution types and score functions considered throughout this thesis, especially given the fact that the task execution time decision variables must be re-optimized within the stochastic metrics for each realization of the random variables. These importance sampling methods hold promise, however, and can really improve real-time algorithm convergence, motivating further research in this area.

The robust planning algorithms proposed in this thesis relied on models of the parameter uncertainty being available. Given observations of the uncertain planning parameters, several estimation and inference techniques can be used to create probabilistic models that

can be leveraged within the planning framework. As new data is acquired however, these models need to be updated to ensure that the planner predictions account for the latest knowledge provided by these new measurements. Online learning algorithms have been extensively explored in the literature, and the integration of robust planning and model learning, especially in distributed and dynamic environments, remains an active area of research [180]. Several recent approaches have explored the use of Nonparametric Bayesian models as powerful frameworks to capture unknown system dynamics and behavior models. These approaches seem particularly useful for representing complex unknown environments since they do not require prior knowledge about the number of system modes (e.g. operator behavior modes, vehicle types, target intents, etc), but as new modes are discovered the models are automatically updated to include them. Furthermore, several of these Nonparametric Bayesian approaches work well with limited amounts of data and are able to generalize the models to predict performance over unexplored regions of the model, as well as provide quantification of the uncertainty over the different model regions [108, 178]. This flexible and modular structure is characteristic of Nonparametric Bayesian models making them very well suited to perform inference in unknown and uncertain environments. A few examples of Nonparametric Bayesian models include the Hierarchical Dirichlet Process over Hidden Markov Models (HDP-HMM) [79–81], and the Dirichlet Process over Gaussian Processes (DPGP) approach, presented in [108]. The HDP-HMM algorithm applies a hierarchical Dirichlet process structure to learn the modes and parameters of the underlying HMM model that represents the system of interest. The algorithm is applied to an example problem showing its usefulness for tracking a maneuvering target. The DPGP algorithm consists of a Dirichlet Process (DP) applied over a variable set of Gaussian Processes (GPs) [108]. The DP is used to update the number of modes and GP parameters as more data becomes available. The algorithm is used to predict trajectories of vehicles in an urban environment using example trajectory data for taxi services. These flexible Nonparametric Bayesian models could be leveraged within a robust planning framework to improve mission performance, especially given dynamic mission scenarios.

Given that the models used within the planning framework are derived from the available data, and are thus subject to the uncertainty and accuracy associated with the data set, an active area of research involves developing information-rich planning strategies that aim to explicitly reduce parameter uncertainty through the acquisition of higher quality

data that maximizes information content [39, 168]. An example of planning with the specific object of reducing uncertainty is provided in Appendix B, where we have extended the distributed CBBA framework to enable information-rich task allocation. In this extension, obtaining higher quality data with higher information content is an explicit goal in the planning process, which is balanced alongside the other mission objectives. The approach quantifies the predicted uncertainty reduction by computing the Fisher Information associated with different vehicle trajectories and sensing locations [168]. In joint work with Cornell University and multiple authors (see Appendix B for details), this information-rich CBBA planning algorithm was embedded into a larger distributed information-based control architecture (Section B.3.1), presenting a novel planning and estimation framework, where the goal of maximizing information was a primary objective for each of the algorithms at every step. The proposed framework was validated through a set of real-time experiments at Cornell University, involving a human-robot team performing a multi-target search mission, demonstrating the viability of the approach [169].

Appendix A

Derivations of Agent Risk Allocation Strategies

The heuristic risk allocation methods employed in this work use the expression provided by Eq. (6.8) to determine how to set the agent risks given the mission risk. For convenience, the risk equation derived in Section 6.2, Eq. (6.8), is repeated below in Eq. (A.1),

$$\sum_{i=1}^{N_a} \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i) = \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon) \tag{A.1}$$

This Appendix shows how the different risk allocation methods discussed in Section 6.3.1 can be derived from Eq. A.1.

A.1 Homogeneous Agent Risk Allocation Strategies

The first case considered is for teams of homogeneous agents, where all agents in the team have similar planning parameters and underlying distributions. The heuristic strategies employed in this case assume that the distributions of the agent scores, $\mathbf{f}(\mathbf{z}_i)$, are all identical, and that the risk values ϵ_i will be the same for all agents. Using these assumptions, Eq. (A.1) reduces to

$$N_a \; \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i) = \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)$$

where the identical agent risks are given by

$$\epsilon_i = \mathbf{F}_{\mathbf{z}_i}\left(\frac{1}{N_a}\mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)\right) \tag{A.2}$$

The expression in Eq. A.2 can be used to describe any homogeneous team, however, specifying the mission distribution $\mathbf{f}(\mathbf{z})$ may be difficult given certain agent score distributions $\mathbf{f}(\mathbf{z}_i)$. In this work, we invoke the Central Limit Theorem, and use a Gaussian distribution to approximate the mission score (sum of agent score random variables). Using this assumption gives,

$$\begin{aligned} \mathbf{z} &\sim \mathcal{N}\big(\mu, \sigma^2\big) \\ \mu &= \sum_{i=1}^{N_a} \mu_i \\ \sigma^2 &= \sum_{i=1}^{N_a} \sigma_i^2 \end{aligned}$$

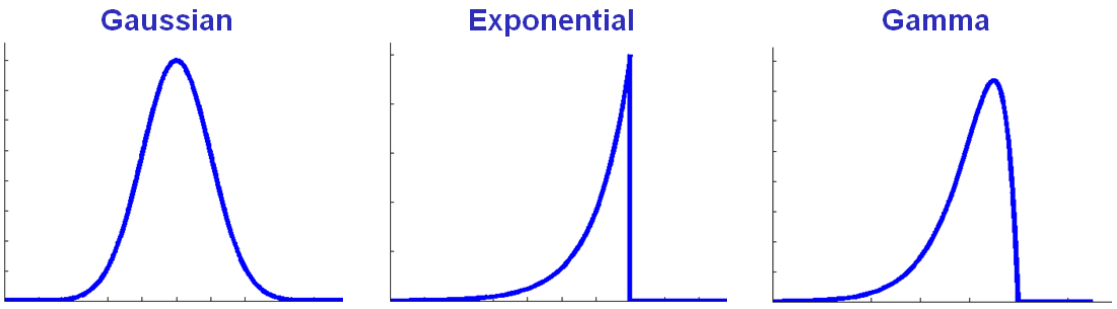
where the mean and variance of the mission distribution are given by the sum of the means and variances of the agent distributions respectively. Given identical agent distributions $\mathbf{f}(\mathbf{z}_i))$ with mean μ_i and variance σ_i^2, the mission score distribution becomes $\mathbf{z} \sim \mathcal{N}\big(N_a\mu_i, N_a\sigma_i^2\big)$. For Gaussian random variables, the CDF and inverse CDF are given by the following expressions,

$$\begin{aligned} \mathbf{F}_X(x) &= \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sqrt{2\sigma^2}}\right)\right) \\ \mathbf{F}_X^{-1}(\epsilon) &= \mu + \sqrt{2\sigma^2}\operatorname{erf}^{-1}(2\epsilon - 1) \end{aligned} \tag{A.3}$$

Using this Gaussian approximation for the mission score distribution, Eq. (A.2) can be written as

$$\begin{aligned} \epsilon_i &= \mathbf{F}_{\mathbf{z}_i}\left(\frac{1}{N_a}\left(N_a\mu_i + \sqrt{2N_a\sigma_i^2}\operatorname{erf}^{-1}(2\epsilon - 1)\right)\right) \\ &= \mathbf{F}_{\mathbf{z}_i}\left(\mu_i + \sqrt{\frac{2}{N_a}}\sigma_i \operatorname{erf}^{-1}(2\epsilon - 1)\right) \end{aligned} \tag{A.4}$$

The expression provided in Eq. (A.4) can be used with many different forms of the agent distributions $\mathbf{f}(\mathbf{z}_i)$. In this thesis, we explored three different agent score distribution forms



(a) Gaussian Risk (b) Exponential Risk (c) Gamma Risk

Figure A-1: Agent score distributions used in the three different homogeneous risk heuristics.

for homogeneous agents based on Gaussian, exponential and gamma distributions for $\mathbf{f}(\mathbf{z}_i)$. Illustrations of the distribution shapes for these three different homogeneous risk heuristics are provided in Figure A-1. The intuition behind using the two nonsymmetric distributions shown in the exponential and gamma cases of Figure A-1 is that, for the types of time-critical mission scenarios considered throughout this thesis, the score distributions for agents tended to have probability masses clustered around maximum task rewards and diminishing probabilities associated with obtaining lower scores. This was because arriving at a task on time or early resulted in agents receiving full task scores, whereas arriving late (but within the window of validity) resulted in exponentially decreasing task scores. The derivations for these three homogeneous risk allocation strategies are provided in the next sections.

A.1.1 Gaussian Risk Allocation Heuristic

In this risk allocation strategy, the agent scores are assumed to be Gaussian random variables $\mathbf{z}_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ with mean μ_i and variance σ_i^2. Replacing $\mathbf{F}_{\mathbf{z}_i}$ in Eq. (A.4) with a Gaussian CDF (see Eq. (A.3)) gives the following derivation for the agent risks,

$$\begin{aligned} \epsilon_i &= \mathbf{F}_{\mathbf{z}_i}\left(\mu_i + \sqrt{\frac{2}{N_a}}\sigma_i \operatorname{erf}^{-1}(2\epsilon - 1)\right) \\ &= \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{\mu_i + \sqrt{\frac{2}{N_a}}\sigma_i \operatorname{erf}^{-1}(2\epsilon - 1) - \mu_i}{\sqrt{2}\sigma_i}\right)\right) \\ &= \frac{1}{2}\left(1 + \operatorname{erf}\left(\sqrt{\frac{1}{N_a}} \operatorname{erf}^{-1}(2\epsilon - 1)\right)\right) \end{aligned} \tag{A.5}$$

The expression in Eq. (A.5) provides a way to set the agent risks ϵ_i given the mission risk ϵ which will be accurate when the agent scores approach Gaussian distributions. In many scenarios of interest, however, agents often have non-symmetric score distributions. In particular, for the types of time-varying score functions considered throughout this thesis (see Section 5.3), the maximum task score is obtained when agents arrive at tasks early or on time, and the score diminishes when agents are late to their tasks. This leads to agent scores with hybrid distributions containing both discrete and continuous components (since there is a delta function at the maximum score value), which are difficult to handle analytically within the risk allocation framework (as an example, see Figure 6-7(b)). In particular, the risk equivalence in Eq. (6.8) was derived assuming that CDFs for the distributions were invertible (e.g. continuous random variables). In this work, we approximate these hybrid distributions for agent scores using non-symmetric continuous distributions. In particular, two strategies are explored involving exponential distributions and gamma distributions. As shown in Figure 6-7(b), the agent score distributions resemble exponential or gamma distributions flipped about the vertical axis and shifted over by some amount. This type of transformation (flipping and sliding), involves applying a linear transformation to the random variable, $Y = aX + b$, where $a = -1$ (flip) and b is some quantity corresponding to the shift. For linear transformations of random variables, given $Y = aX + b$ and the original CDF $\mathbf{F}_X(x)$, the CDF of the transformed random variable is computed using,

$$\mathbf{F}_Y(y) = \begin{cases} \mathbf{F}_X\left(\frac{y-b}{a}\right), & a > 0 \\ 1 - \mathbf{F}_X\left(\frac{y-b}{a}\right), & a < 0 \end{cases} \tag{A.6}$$

The exponential and gamma risk allocation strategies proposed in this thesis make use of these transform equations and are explained in detail next.

A.1.2 Exponential Risk Allocation Heuristic

For a random variable distributed according to an exponential distribution, with parameter λ, the CDF, mean, and variance are given by the following expressions,

$$\begin{aligned} \mathbf{F}_X(x) &= 1 - e^{-\lambda x} \\ \mu_X &= \frac{1}{\lambda} \end{aligned} \tag{A.7}$$

$$\sigma_X^2 = \frac{1}{\lambda^2}$$

Since we are interested in agent score distributions that resemble an exponential random variable flipped about the vertical axis and shifted by some amount, we apply the transform described above with $a = -1$ and some shift value b, where the CDF transform expression is given by Eq. (A.6) using $a < 0$. The CDF, mean, and variance for this transformed exponential random variable are given by the following expressions,

$$\begin{aligned} \mathbf{F}_Y(y) &= e^{-\lambda(b-y)} \\ \mu_Y &= \frac{-1}{\lambda} + b \\ \sigma_Y^2 &= \frac{1}{\lambda^2} \end{aligned} \tag{A.8}$$

Using the CDF, mean, and variance of this transformed random variable in Eq. (A.4) gives the following expression for agent risks,

$$\begin{aligned} \epsilon_i &= \mathbf{F}_{\mathbf{z}_i}\left(\mu_i + \sqrt{\frac{2}{N_a}}\sigma_i \operatorname{erf}^{-1}(2\epsilon - 1)\right) \\ &= e^{-\lambda\left(b-\mu_i-\sqrt{\frac{2}{N_a}}\sigma_i \operatorname{erf}^{-1}(2\epsilon-1)\right)} \\ &= e^{-\lambda\left(b+\frac{1}{\lambda}-b-\sqrt{\frac{2}{N_a}}\left(\frac{1}{\lambda}\right) \operatorname{erf}^{-1}(2\epsilon-1)\right)} \\ &= e^{-\left(1-\sqrt{\frac{2}{N_a}} \operatorname{erf}^{-1}(2\epsilon-1)\right)} \end{aligned} \tag{A.9}$$

Although exponential random variables are non-symmetric and do capture the types of agent scores observed in this thesis, the shape of the distribution is fixed (the *scale* can be controlled through the parameter λ but the *shape* is fixed). In some situations, it is preferable to use a gamma distribution instead, since it provides more control over the shape of the distribution as well as the scale. The next section describes a risk allocation strategy using gamma distributions.

A.1.3 Gamma Risk Allocation Heuristic

A very similar strategy to the one used in the exponential heuristic is used to derive the gamma risk allocation heuristic. For a random variable distributed according to a gamma distribution, with parameters k and θ (controlling the shape and scale respectively), the

CDF, mean, and variance are given by the following expressions,

$$\begin{aligned} \mathbf{F}_X(x) &= \frac{1}{\Gamma(k)}\gamma\left(k, \frac{x}{\theta}\right) \\ \mu_X &= k\theta \\ \sigma_X^2 &= k\theta^2 \end{aligned} \tag{A.10}$$

where $\Gamma(k)$ is the gamma function and $\gamma(k, x)$ is the incomplete gamma function given by,

$$\begin{aligned} \Gamma(k) &= \int_0^\infty e^{-t}t^{k-1}dt \\ \gamma(k, x) &= \int_0^x e^{-t}t^{k-1}dt \end{aligned}$$

Since we are interested in agent score distributions that resemble a gamma random variable flipped about the vertical axis and shifted by some amount, we apply a linear transform with $a = -1$ and some shift value b, where the CDF transform expression is given by Eq. (A.6) using $a < 0$. The CDF, mean, and variance for this transformed gamma random variable are given by the following expressions,

$$\begin{aligned} \mathbf{F}_Y(y) &= 1 - \frac{1}{\Gamma(k)}\gamma\left(k, \frac{b-y}{\theta}\right) \\ \mu_Y &= -k\theta + b \\ \sigma_Y^2 &= k\theta^2 \end{aligned} \tag{A.11}$$

Using the CDF, mean, and variance of this transformed random variable in Eq. (A.4) gives the following expression for agent risks,

$$\begin{aligned} \epsilon_i &= \mathbf{F}_{\mathbf{z}_i}\left(\mu_i + \sqrt{\frac{2}{N_a}}\sigma_i \operatorname{erf}^{-1}(2\epsilon - 1)\right) \\ &= 1 - \frac{1}{\Gamma(k)}\gamma\left(k, \frac{1}{\theta}\left(b - \mu_i - \sqrt{\frac{2}{N_a}}\sigma_i \operatorname{erf}^{-1}(2\epsilon - 1)\right)\right) \\ &= 1 - \frac{1}{\Gamma(k)}\gamma\left(k, \frac{1}{\theta}\left(b + k\theta - b - \sqrt{\frac{2k}{N_a}}\theta \operatorname{erf}^{-1}(2\epsilon - 1)\right)\right) \\ &= 1 - \frac{1}{\Gamma(k)}\gamma\left(k, k - \sqrt{\frac{2k}{N_a}} \operatorname{erf}^{-1}(2\epsilon - 1)\right) \end{aligned} \tag{A.12}$$

In the case where $k = 1$ the gamma distribution and the exponential distribution are equivalent (where θ is related to λ by $\theta = 1/\lambda$), and thus the risk allocation heuristics for gamma and for exponential random variables return the same values for ϵ_i.

The risk expressions for the three homogeneous risk allocation heuristics presented here are summarized in Eq. A.13,

$$\begin{aligned} Gaussian : \epsilon_i &= \frac{1}{2}\left(1 + \operatorname{erf}\left(\sqrt{\frac{1}{N_a}}\operatorname{erf}^{-1}(2\epsilon - 1)\right)\right) \\ Exponential : \epsilon_i &= e^{-\left(1-\sqrt{\frac{2}{N_a}}\operatorname{erf}^{-1}(2\epsilon-1)\right)} \\ Gamma : \epsilon_i &= 1 - \frac{1}{\Gamma(k)}\gamma\left(k, k - \sqrt{\frac{2k}{N_a}}\operatorname{erf}^{-1}(2\epsilon - 1)\right) \end{aligned} \tag{A.13}$$

Note that in all of these homogeneous risk allocation expressions, the individual agent risk values are not affected by the shift and scale parameters of the distributions (e.g. μ and σ in the Gaussian case, b and λ in the exponential case, and b and θ in the gamma case). The heuristic risk allocation remains constant regardless of the means and variances of the underlying distributions. Since the agent score distributions are convolved to give the mission distribution, means and variances cancel out in Eq. (6.8) since they appear on both sides of the equation in equal magnitudes. The intuition behind this observation is that the risk allocation process is affected by the geometry and *shape* associated with the distributions (particularly the tails), and not the scale and shift parameters or the distributions themselves.

A.2 Heterogeneous Agent Risk Allocation Strategies

Setting the risk values for heterogeneous agents is a bit more complicated, since the assumptions made in Eq. (A.2) regarding identical agent distributions and identical risk values may no longer hold. For general problems, Eq. (A.1) will have infinite possible combinations of ϵ_i as valid solutions for a given specific value of ϵ, therefore specifying different individual agent risks becomes difficult. There are two main goals associated with allocating risks amongst the agents. The first goal is that the risks given to individual agents should be such that the global mission risk level is adequately captured by the team. This was the purpose of Eq. (A.1) which identified a relationship between mission risk and agent risks given available plan distributions. The second goal is that the risks allocated to the agents

should encourage agents to pick "better" plans, such that the chance-constrained mission score $\mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)$ be as high as possible. This involves finding a distribution for the mission score $\mathbf{z}$ that maximizes $\mathbf{F}_{\mathbf{z}}^{-1}(\epsilon)$, however, $\mathbf{f}(\mathbf{z})$ is a function of the agent score distributions $\mathbf{f}(\mathbf{z}_i)$ (e.g. a convolution of these agent distributions if the agents are independent), and the distributions $\mathbf{f}(\mathbf{z}_i)$ are in turn functions of the risk levels ϵ_i and of the inner workings of the planner (which are hard to predict). This severe coupling makes the goal of optimizing the ϵ_i allotments to achieve the best plan very difficult. Another issue in distributed planning environments, is that the agents must be able to select their own values ϵ_i given statistics about the mission and the other agents, or must be able to share information with each other (e.g. distributions, moments, or even ϵ_i allocations) to converge on a consistent allocation of the risk levels ϵ_i.

With these issues in mind, this thesis considers a few different heuristic strategies to allocate risks amongst agents given a heterogeneous team. The first heuristic considered assumes that all agents are given identical risk values ϵ_i (note that this does *not* imply that the agents have identical distributions). Invoking the Central Limit Theorem again, the right hand side of Eq. (A.1) is assumed to be Gaussian, where the mission score distribution is given by,

$$\begin{aligned} \mathbf{z} &\sim \mathcal{N}\big(\mu, \sigma^2\big) \\ \mu &= \sum_{i=1}^{N_a} \mu_i \\ \sigma^2 &= \sum_{i=1}^{N_a} \sigma_i^2 \end{aligned}$$

and Eq. (A.1) can be re-written as,

$$\sum_{i=1}^{N_a} \mathbf{F}_{\mathbf{z}_i}^{-1}(\epsilon_i) = \mu + \sqrt{2\sigma^2}\,\mathrm{erf}^{-1}(2\epsilon - 1) \tag{A.14}$$

Since the agent distributions are possibly all different, the left side of Eq. (A.1) is still difficult to compute, depending on the particular CDFs of the agent score distributions. In this work, we assume that agent distributions are also Gaussian, $\mathbf{z}_i \sim \mathcal{N}\big(\mu_i, \sigma_i^2\big)$, where the agent means and variances are assumed to be different since the team is heterogeneous. Using identical risk values and heterogeneous Gaussian distributions in Eq. (A.14), an

analytic expression for the agent risks ϵ_i can be obtained as follows,

$$\begin{aligned}
\sum_{i=1}^{N_a}\left(\mu_i + \sqrt{2\sigma_i^2}\,\mathrm{erf}^{-1}(2\epsilon_i - 1)\right) &= \sum_{i=1}^{N_a}\mu_i + \sqrt{2\sum_{i=1}^{N_a}\sigma_i^2}\,\mathrm{erf}^{-1}(2\epsilon - 1) \qquad \text{(A.15)}\\
\mathrm{erf}^{-1}(2\epsilon_i - 1)\left(\sum_{i=1}^{N_a}\sqrt{\sigma_i^2}\right) &= \sqrt{\sum_{i=1}^{N_a}\sigma_i^2}\,\mathrm{erf}^{-1}(2\epsilon - 1)\\
\mathrm{erf}^{-1}(2\epsilon_i - 1) &= \left(\frac{\sqrt{\sum_{i=1}^{N_a}\sigma_i^2}}{\sum_{i=1}^{N_a}\sqrt{\sigma_i^2}}\right)\mathrm{erf}^{-1}(2\epsilon - 1)
\end{aligned}$$

where solving for the risk value ϵ_i gives,

$$\begin{aligned}
\epsilon_i &= \frac{1}{2}\left(1 + \mathrm{erf}\big(\mathbf{H}\,\mathrm{erf}^{-1}(2\epsilon - 1)\big)\right) \qquad \text{(A.16)}\\
\mathbf{H} &= \left(\frac{\sqrt{\sum_{i=1}^{N_a}\sigma_i^2}}{\sum_{i=1}^{N_a}\sqrt{\sigma_i^2}}\right)
\end{aligned}$$

with the constant value $\mathbf{H}$ representing the team heterogeneity with regards to variance in agents' scores. This expression has several interesting properties. Firstly, the agent risk values for the Gaussian case do not depend on the means of the agent distributions or mission distribution, they only depend on the variances. This is similar to the observation made about the homogeneous risk allocation strategies, where the means of the distributions and scale parameters did not affect the risk allocation. However, in the heterogeneous case, the *relative* scale parameters do affect the risk allocation, as captured by the constant $\mathbf{H}$ in Eq. (A.16). Given the expression in Eq. (A.16), if the agents are homogeneous (with identical distributions), then $\mathbf{H} = 1/\sqrt{N_a}$ and the expression is equivalent to the homogeneous Gaussian risk allocation presented in Eq. (A.13). On the other hand, if the entire mission distribution comes from only 1 agent's contribution (all other agents are deterministic with no variance), then $\mathbf{H} = 1$ and $\epsilon_i = \epsilon$ as expected. This shows that team heterogeneity can be represented via the parameter $\mathbf{H}$, which is a function of N_a and of the relative scales of the agent distributions with respect to one another. The range of $\mathbf{H}$ is given by $\mathbf{H} \in \left[\frac{1}{\sqrt{N_a}}, 1\right]$. A major advantage of using this heuristic versus more complex allocations between heterogeneous agents, is that agents can select a number within the range of $\mathbf{H}$ that roughly represents how heterogeneous the team is (possibly performing consensus on

this number), and then use $\mathbf{H}$ to compute ϵ_i individually. This is significantly faster than coming to consensus on a consistent allocation of the individual parameters ϵ_i.

An alternate heuristic risk allocation strategy considered involves assigning different risk values ϵ_i to different types of agents, where agents of the same type would be assigned identical values of ϵ_i. For example, consider a scenario with 2 types of agents, and with equal numbers of each type of agent. For this scenario, Eq. (A.1) becomes,

$$\frac{N_a}{2}\mathbf{F}_{\mathbf{z}_1}^{-1}(\epsilon_1) + \frac{N_a}{2}\mathbf{F}_{\mathbf{z}_2}^{-1}(\epsilon_2) = \mathbf{F}_{\mathbf{z}}^{-1}(\epsilon) \tag{A.17}$$

where the distributions and risks for each different agent type k are given by $\mathbf{z}_k$ and ϵ_k. Assuming Gaussian agent scores, and Gaussian mission scores as in the previous risk allocation strategy, Eq. (A.17) simplifies to the following expression,

$$\frac{\sigma_1}{\sqrt{\sigma_1^2+\sigma_2^2}}\,\mathrm{erf}^{-1}(2\epsilon_1-1) + \frac{\sigma_2}{\sqrt{\sigma_1^2+\sigma_2^2}}\,\mathrm{erf}^{-1}(2\epsilon_2-1) = \sqrt{\frac{2}{N_a}}\,\mathrm{erf}^{-1}(2\epsilon-1) \tag{A.18}$$

Similar expressions can be derived given 3 or more types of agents. In Eq. (A.18), each of the terms on the left hand side include a scaling parameter that is proportional to the standard deviation for that agent type (normalized by the standard deviation of the mission). The right hand side of Eq. (A.18) includes the number of agents N_a and is typically a function of the number of agent types as well. Given the expression in Eq. (A.18), a key question involves deciding how to partition the risk amongst the agent types. This can be accomplished by splitting the right hand side of Eq. (A.18) into shares, and then solving for ϵ_k for each agent type k (where the agent risks would be set using $\epsilon_i = \epsilon_k$ for agents belonging to type k). It is not obvious, however, how these shares should be divided amongst the agent types. In this thesis we consider two cases, one involving equal shares, and one setting shares proportional to the standard deviation of the agent type σ_k.

The first strategy, which uses equal shares, divides the right hand side of Eq. (A.18) into two equal parts, giving the following expression for each group k,

$$\frac{\sigma_k}{\sqrt{\sigma_1^2+\sigma_2^2}}\,\mathrm{erf}^{-1}(2\epsilon_k-1) = \frac{1}{2}\sqrt{\frac{2}{N_a}}\,\mathrm{erf}^{-1}(2\epsilon-1) \tag{A.19}$$

Each agent type k must then solve for ϵ_k as follows,

$$\epsilon_k = \frac{1}{2}\left(1 + \text{ erf}\left(\left(\frac{\sqrt{\sigma_1^2+\sigma_2^2}}{2\sigma_k}\sqrt{\frac{2}{N_a}}\right)\text{ erf}^{-1}(2\epsilon-1)\right)\right) \tag{A.20}$$

The quantity preceding the inverse error function in Eq. (A.20) can be thought of as a scaling constant $\mathbf{H}_k$ to represent agent heterogeneity, where

$$\mathbf{H}_k = \left(\frac{\sqrt{\sigma_1^2+\sigma_2^2}}{2\sigma_k}\sqrt{\frac{2}{N_a}}\right) \tag{A.21}$$

in Eq. (A.20). Different values of $\mathbf{H}_k$ will lead to different risk allocations ϵ_k, and again it is not immediately obvious how to partition the shares (how to set $\mathbf{H}_k$) to get an allocation of ϵ_i's for all agents that optimizes the chance-constrained mission score.

The second strategy used in this thesis assumes that the shares agents get are proportional to their standard deviation, thus the right hand side of Eq. (A.18) is divided into shares of size $\sigma_k / \sum_k \sigma_k$. Given this division, each group has the following expression,

$$\frac{\sigma_k}{\sqrt{\sigma_1^2+\sigma_2^2}}\text{ erf}^{-1}(2\epsilon_k-1) = \frac{\sigma_k}{\sigma_1+\sigma_2}\sqrt{\frac{2}{N_a}}\text{ erf}^{-1}(2\epsilon-1) \tag{A.22}$$

and solving for ϵ_k gives,

$$\epsilon_k = \frac{1}{2}\left(1 + \text{ erf}\left(\left(\frac{\sqrt{\sigma_1^2+\sigma_2^2}}{\sigma_1+\sigma_2}\sqrt{\frac{2}{N_a}}\right)\text{ erf}^{-1}(2\epsilon-1)\right)\right) \tag{A.23}$$

where the constant $\mathbf{H}_k$ becomes

$$\mathbf{H}_k = \left(\frac{\sqrt{\sigma_1^2+\sigma_2^2}}{\sigma_1+\sigma_2}\sqrt{\frac{2}{N_a}}\right) \tag{A.24}$$

Note that in this case, the constant $\mathbf{H}_k$ is not explicitly dependent on the individual parameter σ_k anymore, but rather considers statistics over the variances for all agent types. As a result, $\mathbf{H}_k$ will be constant for all agent types k and therefore the risk values ϵ_k will all be the same, leading to equal risks for all agents in the team (but still capturing the heterogeneity associated with the different variances for agent scores). It is shown in Section 6.4 that this last strategy, where risks are equal for all agents, performs significantly better than the strategy shown in Eq. (A.20). This is because by balancing the risks more

evenly throughout the team, no agent can take on its extreme plan values (very deterministic taking no risk at all, or taking too much risk and not considering how it might affect the mission as a whole), which increases the performance of the team. Furthermore, the heuristic strategy employed in Eq. (A.16), which also assigned equal risks to all the agents, also achieved high performance which was significantly better than the strategy shown in Eq. (A.20), and on par with the strategy of Eq. (A.23). On closer inspection, the value for $\mathbf{H}_k$ in Eq. (A.23) looks very similar to that of $\mathbf{H}$ in Eq. (A.16), explaining why the performance of both heuristics was similar, since they both capture the same relative scaling effects associated with the heterogeneous agent variances.

Appendix B

Distributed Information-Rich Planning and Hybrid Sensor Fusion

Chapters 5 and 6 of this thesis presented methods for planning in distributed environments, with particular focus on how to plan robustly given uncertainty in the environment. Methods for embedding distribution models into the planner were described with the objective of hedging against the uncertainty in the environment to improve planning performance. A key consideration, however, is that the plans obtained, even using robust planning strategies, are directly affected by the distribution models available and the amount of uncertainty associated with the planning parameters. A more active approach to handling uncertainty within the planner involves explicitly accounting for uncertainty reduction through information-rich planning strategies. The basic notion is that, by actively controlling the measurement process (e.g. sensor locations, vehicle trajectories), the uncertainty associated with the planner models can be further reduced through the collection of higher quality data that maximizes information content.

This appendix presents a novel planning and estimation architecture leveraging CBBA, where the goal of maximizing information is a primary objective for each of the algorithms at every step, producing a cohesive framework that strategically addresses the main mission objectives. The unified system architecture consists of: distributed task planning using CBBA extended to include an information-based objective criteria, decentralized information-based trajectory planning using the IRRT algorithm [133], and a hybrid Bayesian information fusion algorithm that uses Gaussian mixture models to represent target position [4]. The proposed framework is validated through a set of real-time experiments involving a human-

robot team performing a multi-target search mission, demonstrating the viability of the approach.

The work presented in this appendix consists of joint efforts with Cornell University, with several participating authors, including: Nisar Ahmed (Cornell), Brandon Luders (MIT), Daniel Levine (MIT), Eric Sample (Cornell), Tauhira Hoossainy (Cornell), Danelle Shah (Cornell), Mark Campbell (Cornell), and Jonathan P. How (MIT). This research was supported in part by MURI FA9550-08-1-0356 and a National Science Foundation Graduate Research Fellowship. For more details the reader is referred to [169].

B.1 Introduction

Modern day mission operations often involve large teams of networked agents, with heterogeneous capabilities, interacting together to perform the requisite mission tasks. Such missions typically involve executing several different types of task at once, such as intelligence, surveillance, and reconnaissance (ISR), target classification, rescue operations, scientific exploration, and security monitoring [1, 2]. Furthermore, within the heterogeneous team, some specialized agents are better suited to handle certain types of tasks than others. For example, autonomous air and ground vehicles equipped with video can be used to perform target search and track, human operators can be used for target classification tasks, and ground teams can be deployed to perform rescue operations or engage targets.

Ensuring proper coordination and collaboration between agents in the team is crucial to efficient and successful mission execution. As a result, there has been increasing interest in exploring efficient methods to plan for mixed human-robot teams for various types of missions. Furthermore, the advancement of communication systems, sensors, and embedded technology has significantly increased the value of those solutions that are scalable to larger teams, from dozens to hundreds or even thousands of agents [1, 2]. In such complex systems, care must be taken to balance the resources allocated to primary mission tasks (e.g. search and tracking) and related secondary tasks (e.g. maintenance, monitoring, safety, retrieval, etc).

There are many technical challenges associated with developing algorithms that can effectively coordinate the behavior of such teams. For example, consider a scenario where a team of human operators and autonomous robots is tasked with searching for, tracking,

and classifying unknown targets in an obstacle-filled environment. A key research question is how to efficiently allocate limited agent resources with the objective of minimizing target state uncertainty as quickly as possible, while simultaneously executing required secondary tasks (e.g. vehicle status monitoring, etc). Furthermore, this task assignment process must take into account the challenges associated with the underlying autonomous motion planning and navigation that the agents must perform to successfully accomplish their tasks. For example, the vehicles must be able to autonomously plan trajectories in obstacle-filled and potentially uncertain search environments, minimizing target state uncertainty while also ensuring safety. An additional consideration for this problem is that, given that many disjoint and heterogeneous agents are collaborating to search the environment, it is important to employ efficient information fusion methods, which can be used to effectively combine sensor data acquired by different mobile agents with information from human operators. Since most planning strategies rely on underlying agent models, developing accurate and efficient representations for agents in the team, including human operators, is crucial. In particular, modeling human agents for use in autonomous task allocation and information fusion algorithms remains a challenging problem [205]. Finally, any approach considered should be able to scale with the problem size, characterized by the number of agents and targets, without straining available computational or communication resources.

This work presents an algorithmic approach to tackle task allocation, trajectory planning and information fusion within a *unified framework*, with the objective of reducing uncertainty in the target search and tracking process, while considering the complex constraints associated with realistic human-robot missions. In this novel approach, the goal of maximizing information is a primary objective for each of the algorithms at every step, producing a cohesive framework that strategically addresses the main mission objectives. Both task planning and vehicle path planning are information based, enabling intelligent and efficient cooperative search and track strategies that are balanced alongside other mission objectives. The task allocation and trajectory planning algorithms employed are distributed, making the system scalable to large teams of operators and autonomous agents with diverse potential task sets. Furthermore, the information fusion algorithms presented in this work provide strategies to directly include "soft" inputs from human agents, that can be combined with conventional autonomous sensor information via robust particle filtering algorithms, enabling convenient recursive Bayesian updates for efficient replanning. The

unified task allocation, trajectory planning and information fusion framework is validated in a real-time human-robot multi-target search experiment, demonstrating the viability of the approach.

This paper is organized as follows. Section B.2 defines the problem statement considered by this work. Section B.3 presents the distributed planning and information fusion framework developed to address this problem, including the overall system architecture (Section B.3.1), the information-rich planning algorithms (Section B.3.2), and the Bayesian hybrid data fusion algorithms (Section B.3.3). Indoor target search and track experiments for human-robot teams using the proposed framework are presented and analyzed in Section B.4, followed by concluding remarks in Section B.5. Note that related work is provided throughout the paper, in the corresponding sections.

B.2 Problem Formulation and Background

This work considers the problem of planning for a team of autonomous robotic mobile agents[1] and human operators, tasked with searching for, tracking, and classifying unknown targets in an obstacle-filled dynamic environment. The robotic agents consist of heterogeneous vehicles equipped with onboard computers and a variety of sensors, such as laser range-finders, cameras and visual detection software. The human operators are static and can interact with the robotic agents directly through a computer console. The team's mission is to locate and identify a known number of targets as quickly and accurately as possible in a real-time environment. The details of this search and track problem are described below.

Assume that search region $\mathcal{S} \subseteq \mathbb{R}^3$ contains N static targets with fixed labels $i \in \{1, \ldots, N\}$ and unknown positions $\mathbf{x}_i = [x_i, y_i, z_i]^T$ with respect to some fixed origin (N is known *a priori*). The uncertainty in $\mathbf{x}_i$ is initially modeled by the probability density function (PDF) $p(\mathbf{x}_i)$. This PDF represents any prior beliefs about $\mathbf{x}_i$ (e.g. as obtained from intelligence information, previous experience, or physical considerations). Using the initial target PDFs, $\{p(\mathbf{x}_1), \ldots, p(\mathbf{x}_N)\}$, and a set of observations, $\mathcal{Z}$, acquired by the human-robot team throughout the mission, the primary objective is to detect, identify and localize all N targets in $\mathcal{S}$ as quickly and efficiently as possible. The exact specification of this objective

[1] The framework considered in this paper can be extended to incorporate human-operated mobile agents, though this is not discussed further.

function might include a maximum time limit, a maximum uncertainty covariance for each target, a weighted sum of these factors, or several other considerations (such as specific vehicle constraints).

It is assumed here that each target distribution $p(\mathbf{x}_i)$ is a known M_i-term Gaussian mixture (GM),

$$p(\mathbf{x}_i) = \sum_{m=1}^{M_i} w_{i,m}\, \mathcal{N}(\mu_{i,m}, \Sigma_{i,m}), \tag{B.1}$$

where the parameters $w_{i,m}$, $\mu_{i,m}$, and $\Sigma_{i,m}$ are respectively the weight, mean, and covariance matrix for component m of target i, with $\sum_{m=1}^{M_i} w_{i,m} = 1$. It is well-known that GMs can approximate arbitrarily complex PDFs for suitably chosen M_i and mixing components[44], and are thus quite useful in general estimation problems with significant non-Gaussian uncertainties[119]. At any given time, the aggregated estimate of each target is given by the mean of the distribution which can be computed from the individual modes as

$$\hat{\mathbf{x}}_i = \sum_{m=1}^{M_i} w_{i,m}\, \mu_{i,m}, \tag{B.2}$$

with target covariance given by

$$P_i = \sum_{m=1}^{M_i} w_{i,m} \left[\Sigma_{i,m} + (\mu_{i,m} - \hat{\mathbf{x}}_i)(\mu_{i,m} - \hat{\mathbf{x}}_i)^T\right]. \tag{B.3}$$

The target locations are further assumed to be marginally independent, so that the joint target PDF is given by

$$p(\bar{\mathbf{x}}) = p(\mathbf{x}_1, ..., \mathbf{x}_N) = \prod_{i=1}^{N} p(\mathbf{x}_i). \tag{B.4}$$

If the human-robot team acquires a set of shared target observations $\mathcal{Z}_k$ up to time step k, then the distribution for $\mathbf{x}_i$ can be updated via Bayes' rule as

$$\begin{aligned} p(\mathbf{x}_i|\mathcal{Z}_k) &= \frac{1}{K} p(\mathbf{x}_i) p(\mathcal{Z}_k|\mathbf{x}_i), \\ K &= \int p(\mathbf{x}_i) p(\mathcal{Z}_k|\mathbf{x}_i) d\mathbf{x}_i, \end{aligned} \tag{B.5}$$

where $p(\mathcal{Z}_k|\mathbf{x}_i)$ is the likelihood function for the observations $\mathcal{Z}_k$, and K is a normalizing constant.

In the context of mixed human-robot search teams, the likelihood function, $p(\mathcal{Z}_k|\mathbf{x}_i)$, is composed of several independent models describing how measurements from various sensing platforms are stochastically generated as a function of the underlying target states. For robotic agents, the likelihood function characterizes measurements arising from typical robot sensing platforms, such as cameras and LIDAR. In human-robot search teams, human operators also contribute important target information, particularly with respect to target identification and high-level target behaviors [134], but this information typically has limited usefulness in reducing uncertainty in $\mathbf{x}_i$, since it is either not very related (e.g. target classification), or cannot be properly modeled in $p(\mathcal{Z}_k|\mathbf{x}_i)$ unless the target has been extensively characterized through an *a priori* behavioral model. However, human operator insight is often valuable in guiding search missions, and, in many cases, it is desirable to include these "low-level" observations from operators as "soft inputs" in $\mathcal{Z}_k$ in Equation (B.5), thus allowing human insight to be treated as a sensor that returns continuous or categorical observations of continuous states, such as the target locations [4, 111].

An alternative characterization of the search and track problem described above involves modeling the search mission as an optimal control problem, where the objective is to place the sensing agents on trajectories that maximize the probability of finding the targets over a given time horizon. One strategy to accomplish this is to minimize the uncertainty in the posterior (Equation (B.5)), for example, by using a receding horizon planning strategy that accounts for sensor platform dynamics [47]. For heterogeneous multi-agent search teams, a centralized planning approach with a shared information set could be used in the optimization, but such methods usually scale poorly with the size of the search area, target population, and the number of agents. Recent work [77] considers how to perform decentralized target search in two dimensions, via a discretized representation; however, this approach also scales poorly in three dimensions and with increasing problem sizes, as well as with other realistic constraints such as target dynamics and communication constraints.

In this work, an information-based approach is employed to address the search and track problem at both the task assignment and trajectory planning levels. The solution methodologies do *not* require the discretization of the search space, although the environment is assumed to be bounded and non-convex. The task assignment process determines which

agents are best suited to track which targets given their sensor configurations, current pose, and the prior target estimates provided by the GMs (Section B.3.2). Once the targets are assigned to the respective vehicles, the motion planning algorithm designs information-rich kinodynamically feasible trajectories which traverse this continuous environment while satisfying all state and input constraints [133] (Section B.3.2). The vehicles are assumed to have known dynamics and sensor/detection models (though they may be nonlinear), such that predicted trajectories can be generated deterministically. Reliable pose estimates and environmental obstacle maps are assumed to be available to each agent for convenience, although extensions to uncertain pose and maps are also possible and will be studied in future work. Furthermore, all trajectory planning is decentralized and performed by each vehicle independently; the paths of other agents are assumed unknown, although this information could be shared among the agents. While more efficient sensor fusion can be achieved in such extended search problems using GM representations [216], there has been little prior work on how to effectively embed GMs into the planning framework. The algorithms proposed by this paper incorporate the GM target representations at each level of planning, including task allocation, trajectory planning, and human operator interface. By using computationally efficient algorithms in each of these phases, it is possible for large teams to develop real-time plans which explicitly account for the nature of the target uncertainty at every level.

B.3 Decentralized Planning and Fusion Framework

This section outlines the proposed framework for managing a team of human operators and autonomous vehicles engaged in a generalized target search, tracking, and identification mission. The presented approach consists of three primary algorithmic components: task allocation, trajectory planning, and information fusion. The key contribution of this work is the development of a unified framework which integrates these algorithms, allowing for the explicit consideration of target uncertainty reduction, complex constraints, and secondary objectives (e.g. safety, refueling, etc.) at every level of planning. Section B.3-B.3.1 presents the overall system architecture. Section B.3-B.3.2 reviews the task planning and vehicle path planning algorithms, describing how information gains are directly accounted for in the planning process, enabling the algorithms to balance information collection with other

mission objectives. Finally, Section B.3-B.3.3 presents the hybrid Bayesian fusion strategy, which combines traditional sensor models with low-level categorical human observations of target states.

B.3.1 Proposed Information-based Control Architecture

This section presents the overall system architecture for the types of planning and fusion problems considered in this work, describing the relationship between the individual components. A diagram of the generalized framework is presented in Figure B-1. The main components, as shown in the figure, consist of task allocation, path planning, vehicle and sensor configurations, and state estimation and sensor fusion. The task allocation algorithm receives the latest state estimates of both the vehicles and targets, and uses this information, along with accurate models of the agents and sensors, to determine the assignment of targets to vehicles. These task assignments are then communicated to the individual vehicle path planners. The path planning algorithms design trajectories for the vehicles that minimize the target state uncertainty while considering resource consumption and obstacle avoidance. The vehicles then implement these trajectories, update their pose estimates, and collect observations via their sensors. The individual agent state and sensor data is sent to a state estimation and sensor fusion module that combines all this information to obtain the latest estimates of the agent and target states, along with measures of the estimation uncertainty.

Figure B-2 shows a diagram of the proposed information-rich planning and fusion framework presented in this paper. The task allocation algorithm in the proposed approach consist of the decentralized Consensus-Based-Bundle Algorithm (CBBA) [58] augmented with information metrics, the path planning uses the Information-rich Rapidly-exploring Random Tree (IRRT) [133] algorithm, and the state estimation is performed by a recursive hybrid Bayesian fusion strategy. The hardware platform used to obtain experimental results consisted of a Pioneer rover equipped with cameras (Section B.4). The key principle behind this framework is that task allocation, trajectory planning, and sensor fusion all consider acquiring information and reducing target uncertainty as the primary objectives, creating a unified framework for target tracking that addresses the main mission goals at every level. A secondary advantage is that both the task assignment and trajectory planning are decentralized, as illustrated in Figure B-2, providing a scalable solution methodology which

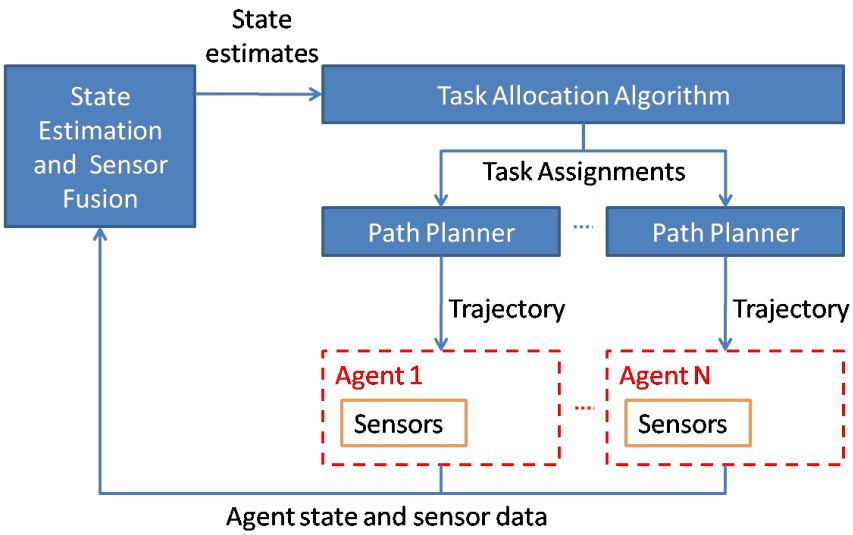


Figure B-1: General system block diagram for proposed planning and fusion framework.

remains computationally tractable as the number of agents and targets increases. An additional contribution illustrated in this framework is the explicit use of human operators in the control and estimation loop, via a human-robot interface (HRI). In this formulation, human operators provide "soft inputs" to the sensor fusion, validating the identity of all potential target detections in addition to other target state information which assists the robots in their search (e.g. these can include fuzzy descriptions of perceived target locations such as 'nearby landmark A' or perceived target behaviors such as 'moving quickly through the side door'). Operators may also be used to handle some secondary tasks, such as monitoring refueling operations or responding to automation failures. The following sections provide further details on these algorithmic system components.

B.3.2 Decentralized Information-Rich Planning

The performance of dynamic search and track missions is typically measured in terms of the efficiency with which the agents involved reduce target estimation uncertainty. However, trajectories that achieve this uncertainty reduction are subject to a complex set of internal and external constraints, including dynamic constraints, environmental restrictions, and sensor limitations. By using the recently-proposed Information-rich Rapidly-exploring Random Tree (IRRT) algorithm [132], a team of agents can quickly identify feasible, uncertainty-reducing paths that explicitly embed the latest target probability distributions, whilst satisfying these constraints. While IRRT is capable of handling multiple

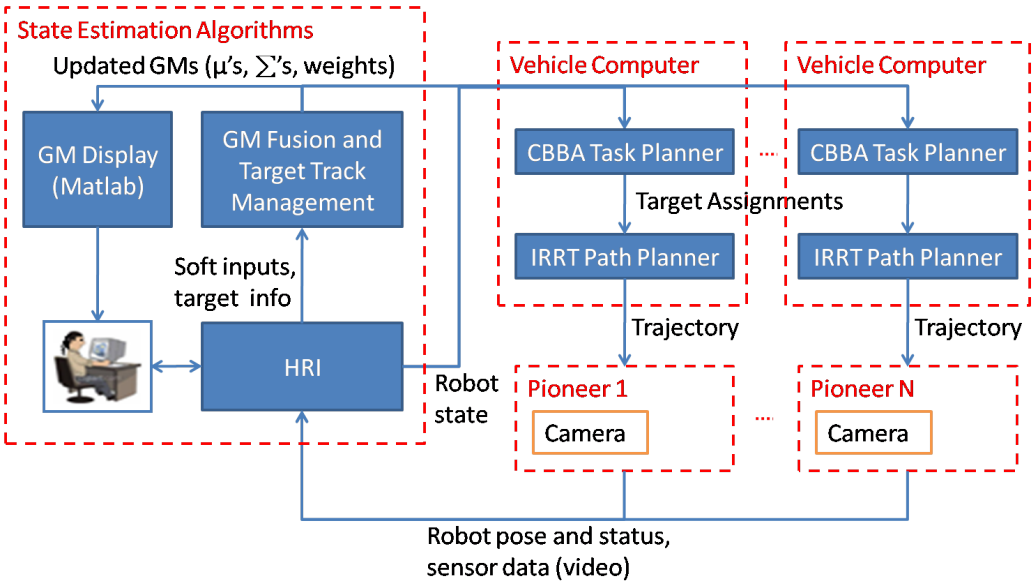
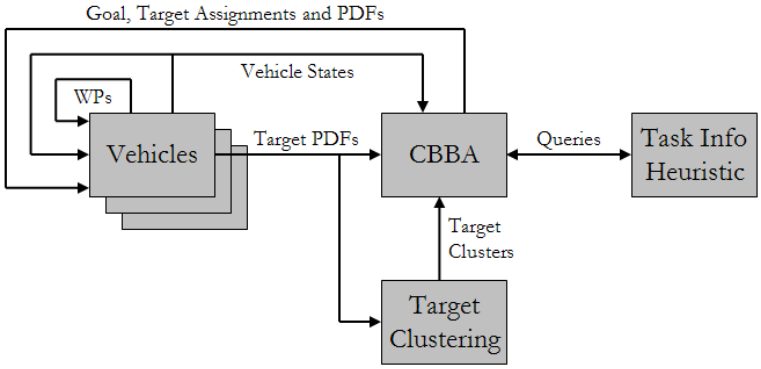


Figure B-2: System block diagram for indoor human-robot target search and track experiment

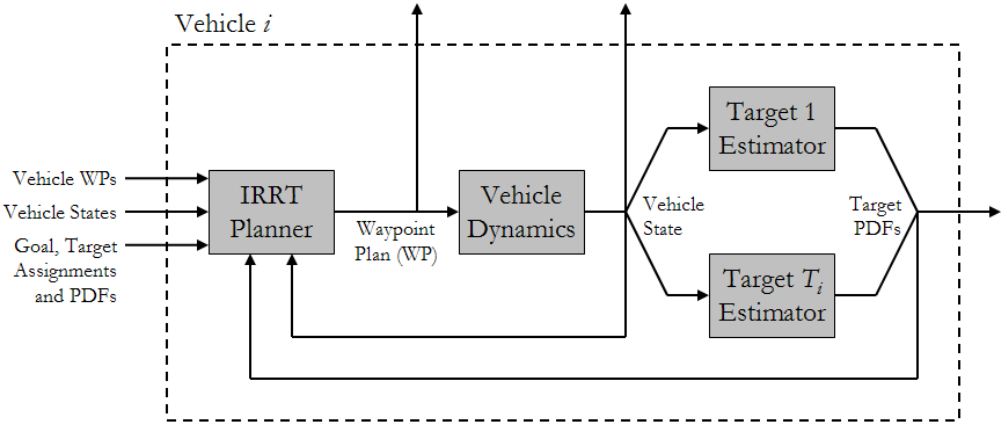
vehicles and targets [133], algorithmic efficiency is lost when considering realistic large-scale ISR missions. Trajectories identified for such scenarios must embed both the vehicle routing problem (in selecting which distant targets to visit) and the constrained sensor problem (in finding a vantage point to view nearby targets), and become computationally intractable as the number of agents and targets increases. By pursuing a distributed approach that partitions the target environment into disjoint tasks and allocates these tasks amongst the agents, the computational burden on the motion planners is reduced. In this work we use a decentralized task allocation algorithm called the Consensus-Based Bundle Algorithm (CBBA) [58] to distribute the targets to the individual agents. The score functions used within the CBBA task allocation framework explicitly account for the information that agents are able to obtain about their assigned targets.

The combination of IRRT+CBBA results in a novel multi-level algorithm which embeds information-rich trajectory planning within a task allocation framework, efficiently assigning targets and planning paths for teams of agents at the mission planning level. This real-time algorithm can leverage networks of mobile sensor agents to perform dynamic task reallocation as target estimates are updated, resulting in improved coordination and collaboration between agents while executing the mission. Figure B-3 shows the proposed IRRT+CBBA architecture, where each vehicle runs an instance of the decentralized CBBA

task allocation algorithm as well as its own IRRT planner. The next sections provide further detail on these two components of the decentralized planning process.



(a) Overall Architecture



(b) Single Vehicle Architecture (leftmost block of Figure B-3(a))

Figure B-3: Block diagrams illustrating the overall CBBA+IRRT integrated architecture.

Decentralized Information-Based Task Allocation

The problem of task allocation has been extensively studied and many different methods have been considered for enabling agents to distribute tasks amongst themselves from a known mission task list [58]. Centralized planners, which rely on agents communicating their state to a central server that generates a plan for the entire fleet, are commonly used in the literature. However, most of these planning architectures require high communication bandwidth, computational resources, and are typically slower to react to changes in local information. Decentralized planning algorithms, where agents make their own plans and communicate amongst themselves, have gained recent popularity, and offer several advan-

tages over centralized planning methods [63, 148]. Many of these decentralized algorithms have to be augmented with consensus algorithms for agents to converge on consistent situational awareness prior to planning [161, 185], a process that can take a significant amount of time and often requires transmitting large amounts of data [7]. A unique decentralized auction algorithm called the Consensus-Based Bundle Algorithm (CBBA) [58] uses a consensus protocol that acts upon the task space only, guaranteeing conflict-free solutions despite possible inconsistencies in situational awareness. CBBA is guaranteed to achieve at least 50% optimality [58], although empirically its performance is shown to be within 93% of the optimal solution [37]. The task selection process of CBBA runs in polynomial time, demonstrating good scalability with increasing numbers of agents and tasks, making it well suited to real-time dynamic environments.

This work uses CBBA to allocate targets to the best suited agents. Figure B-3(a) shows the overall target allocation architecture which is described in this section. Prior to the task allocation process, the targets are grouped into sets using K-means clustering on the target means obtained from the latest target Gaussian mixture estimates. These target sets or "tasks" can then be allocated to the individual agents using CBBA. A key advancement of the CBBA algorithm is a novel information-based scoring framework called the Task Information Heuristic (TIH), which embeds an approximation of the information gain in the assessed value of a target cluster to an agent or team. The TIH consists of selecting a starting location to enter the target cluster[2], followed by a one-step optimization process to find the best information-rich path within the cluster, providing an estimate of the locally optimal information-gathering trajectory. The path optimization involves minimizing the average A-optimality of the individual Fisher Information Matrices for each target [168], and the algorithm continues to extend the path until this average A-optimality is below some uncertainty threshold (or some timeout is reached). The individual target Fisher Information Matrices are initialized using the inverses of the target covariance matrices obtained from the latest target PDFs, thus accounting for the actual acquired information thus far. Finally, the estimated score for the task is computed as the expected acquired information for all targets, minus the fuel resources consumed by following the optimized path. Likewise, the arrival time and task duration are approximated using the agent's

[2]The task start location for each vehicle is determined by computing the closest point on the outer edge of a sphere around the cluster's centroid, whose radius is given by the average cluster spread, with an additional margin to avoid starting inside any target's no-fly zone.

arrival time at the selected start point, and the time required to traverse the optimized path, respectively. Using the estimated scores, task durations, and arrival times, CBBA is able to allocate the tasks to the individual agents producing target lists and expected schedules for each vehicle.

Information-Rich Path Planning

Given the target lists produced by the task allocation process, each agent must plan a trajectory that enables the vehicle to search and track the targets assigned to it as efficiently as possible. Due to its explicit consideration of target uncertainty reduction, this work employs the Information-rich Rapidly-exploring Random Tree (IRRT) algorithm [132, 133]. IRRT uses a closed-loop state prediction in conjunction with sensor models and target prior distributions to simulate a tree of candidate trajectories. Using Fisher information [78], the value of successful measurement poses along each path can be quantified, allowing trajectories to be selected via a trade-off between uncertainty reduction and path duration. As an extension of RRT, the IRRT algorithm is amenable to the general, complex constraint characterizations often encountered in real-world planning problems. This section reviews the IRRT formulation and describes, in particular, how information collection is quantified.

From the perspective of information collection, path quality is a function of the path measurement sequence. And while CL-RRT also enjoys the benefits of smoother path planning on a stabilized vehicle model, it is the added disturbance robustness over open-loop RRT[137] and the associated accurate state prediction that are particularly useful for measurement pose prediction and, therefore, for information-based planning. Because the vehicle's state trajectory is usually simulated with high fidelity, and the result of its prediction is notably accurate, a list of predicted measurement poses $\mathcal{M} = \langle \overline{\mu}_1, \overline{\mu}_2, ..., \overline{\mu}_l \rangle$ can be interpolated for each of many (possibly independent) sensors on the platform. These sensors need not have the same characteristics. Each sensor's list of predicted measurement poses is generated once per node, and thereafter has no need to be updated. Given the most recent modal state estimates $\hat{\mathbf{x}}_{i,m}$ of target i with modes $m \in \{1, \ldots, M_i\}$, each measurement pose $\overline{\mu}_k, k \in \{1, \ldots, l\}$ can be checked against the sensor and environment models to assess visibility. The information for measurements deemed visible is quantified, as described below, and stored in the resulting node n_{new}. Visibility and information quantification of the $\mathcal{M}$ elements may be recomputed as target estimation data is updated.

A myriad of information-theoretic metrics exist to quantify the value of a set of measurements; we use the Fisher Information Matrix (FIM) $J_{\mathbf{Z}}(\mathbf{x})$, which describes the information contained in a set of measurements $\mathbf{z}$ about an estimation process for the vector $\mathbf{x}$. The inverse $J_{\mathbf{Z}}(\mathbf{x})^{-1}$ of the Fisher Information Matrix is exactly the Cramér-Rao Lower Bound (CRLB), a lower bound on the achievable estimation error covariance and thus a quantity to be minimized.[177] A discrete system with linear state transitions and measurements, subject to additive Gaussian white noise, can be modeled as

$$\begin{aligned} \mathbf{x}_{k+1} &= \Phi_{k+1|k}\mathbf{x}_k + \mathbf{w}_k, \\ \mathbf{z}_k &= H_k\mathbf{x}_k + \mathbf{v}_k, \end{aligned} \tag{B.6}$$

where $\Phi_{k+1|k}$ is the state transition matrix, H_k is the linear measurement matrix, $\mathbf{w}_k$ is the process noise, and $\mathbf{v}_k$ is the sensing noise. The process and sensing noises are assumed to be Gaussian, zero-mean and uncorrelated, with covariances given by Q_k and R_k respectively. For such systems, the recursive update equation for the FIM is given by [189]

$$J_{k+1} = (Q_k + \Phi_{k+1|k}J_k^{-1}\Phi_{k+1|k}^T)^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}. \tag{B.7}$$

For stationary targets, $Q_k = 0$ and $\Phi_{k+1|k} = I$ for all k, and the recursion becomes

$$J_{k+1} = J_k + H_{k+1}^T R_{k+1}^{-1} H_{k+1}, \tag{B.8}$$

a particularly convenient form since the FIM in this case is additive, and the information content of a path is just the sum of the FIMs along the path edges. Using this form provides considerable computational savings over planning methods that propagate the covariance, since it does not require the computation of matrix inverses.

The linearity assumption on the observation system can be relaxed by utilizing the linearized FIM as an approximation of the CRLB inverse. Consider systems with discrete measurements $\mathbf{z}$ that are nonlinear in both the target state $\mathbf{x}_i$ and measurement pose μ, and are thus of the form

$$\mathbf{z_k} = \mathbf{h}(\mu_k, \mathbf{x}_i) + \mathbf{v_k}, \tag{B.9}$$

where $\mathbf{v}_k$ is a vector of zero-mean, white Gaussian sequences. The approximate FIM can be formulated by defining H_k to be the Jacobian of the nonlinear measurement function, i.e.,

$$H_k(\overline{\mu}_k, \hat{\mathbf{x}}_i) \triangleq \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mu_k=\overline{\mu}_k, \;\; \mathbf{x}_i=\hat{\mathbf{x}}_i(t)}. \tag{B.10}$$

Note that the assumption of Gaussian noise is retained.

The expected measurement poses $\langle \overline{\mu}_1, \ldots, \overline{\mu}_l \rangle$ can be used in the framework of the FIM to quantify the information content of a particular node in the tree. The approach is to compute FIMs for each target mode separately, approximate a lower bound on the target mixture covariance, and combine the information error from all N targets. The modal FIMs are stored in that node and are used as the initial information conditions for its children. Assuming a Gaussian mixture prior, the recursion is initiated at the root node n_{root} with $J_{root}(\hat{\mathbf{x}}_{i,m}) = P_{i,m}(t)^{-1}$, where $P_{i,m}(t) = \mathbb{E}\left[(\mathbf{x}_{i,m} - \hat{\mathbf{x}}_{i,m}(t))(\mathbf{x}_{i,m} - \hat{\mathbf{x}}_{i,m}(t))^T\right]$ is the error covariance matrix for mode m of target i at time t. For each target i, for each mode m, the FIM $J_b(\hat{\mathbf{x}}_{i,m})$ of a child node n_b is formed by a recursive update from its parent node n_a,

$$J_b(\hat{\mathbf{x}}_{i,m}) = J_a(\hat{\mathbf{x}}_{i,m}) + \sum_{k=1}^{l} \nu(\overline{\mu}_k, \hat{\mathbf{x}}_{i,m}, \hat{\mathcal{E}}) H_k^T(\overline{\mu}_k, \hat{\mathbf{x}}_{i,m}) R_k^{-1} H_k(\overline{\mu}_k, \hat{\mathbf{x}}_{i,m}), \tag{B.11}$$

where l is the number of measurements along the path segment, $\hat{\mathcal{E}}$ is the environment representation, and ν is a binary-valued function capturing the success/occlusion of a measurement. In this way, the tree FIMs are populated and can be recomputed, for example, after the target distributions have been updated.

In the presented approach, the cost associated with information for target i at node n_a is specified as the A-optimality criterion on a lower bound of the mixture covariance, specifically,

$$\mathcal{I}_a(\hat{\mathbf{x}}_i) = \texttt{trace}\left(\sum_{m=1}^{M_i} w_{i,m} J_a^{-1}(\hat{\mathbf{x}}_{i,m})\right). \tag{B.12}$$

The A-optimality criterion has been shown to be better suited than other FIM optimality conditions for the 3D target tracking case[168]. In the multi-target case, convex combina-

Algorithm 10 IRRT, Tree Expansion

1: Take a sample x_{samp} from the environment
2: Identify the nearest node n_{near} using mixture of exploration, optimization, and information heuristics
3: $\overline{x}(t+k) \leftarrow$ final state of n_{near}
4: **while** $\overline{x}(t+k) \in \mathcal{X}_{free}$ **and** $\overline{x}(t+k)$ has not reached x_{samp} **do**
5: Use reference law to generate $\overline{r}(t+k)$
6: Use control law to generate $\overline{u}(t+k)$
7: Use prediction model to simulate $\overline{x}(t+k+1)$
8: $k \leftarrow k+1$
9: **end while**
10: **for** each feasible node n generated **do**
11: Update cost estimates for n
12: Compute simulated measurement poses $n.\mathcal{M}$
13: Compute FIMs using (B.11)
14: Add n to $\mathcal{T}$
15: **end for**

Algorithm 11 IRRT, Execution Loop

1: $t \leftarrow 0$
2: Initialize tree $\mathcal{T}$ with node at $x(0)$
3: **while** $x(t) \neq \mathbf{x}_{goal}$ **do**
4: Update the current state $x(t)$ and target estimates $\hat{x}_i \ \forall i$
5: Propagate the state $x(t)$ by $\Delta t \rightarrow \overline{x}(t+\Delta t)$
6: **while** time remaining for this timestep **do**
7: Expand tree by adding nodes
8: **end while**
9: Update FIMs throughout $\mathcal{T}$ using (B.11)
10: Use information-based cost metric to identify best feasible path, $\{n_{root}, \ldots, n_{select}\}$
11: Apply best feasible path, if one exists
12: $t \leftarrow t + \Delta t$
13: **end while**

tions of the A-optimality costs can be found by summing over the targets,

$$\mathcal{I}_a = \sum_{i=1}^{N} q_i \, \mathcal{I}_a(\hat{\mathbf{x}}_i), \qquad \sum_{i=1}^{N} q_i = 1 \tag{B.13}$$

where the relative weights q_i can be used to bias information collection towards some targets (e.g. mission-critical targets). Summation of the A-optimality costs is consistent with the nature of the multi-objective problem.[3]

The ability to simulate expected measurement poses is used in two ways to extend the CL-RRT algorithm for information gathering. First, these expected measurements are used to bias tree growth toward regions of high information-gain [133] (Algorithm 10)[4]. Second, the vehicle selects paths from the tree that minimize a cost function which explicitly considers information, in addition to path cost and remaining cost-to-go.

Whenever new feasible nodes n_{new} are generated for the tree, the predicted measurement poses $\mathcal{M}$ are stored within the node (line 12). These measurement poses are used to compute the FIMs based on the current target estimates $\hat{\mathbf{x}}_{i,m}$ for all i and m, both when the node is created (line 13) and whenever the best path is selected, as discussed next.

[3]It should be noted that simply summing the FIMs (and not the associated A-optimality costs) over all targets at a given measurement pose is imprudent; for example, two targets with singular FIMs could in their sum form a nonsingular FIM, thereby masking the momentary unobservability of each target's estimation process.

[4]See [137] for more information on existing components.

The IRRT execution loop is presented in Algorithm 11. In the IRRT algorithm, a single, multi-objective cost metric is used (Algorithm 11, line 10), which considers both progress toward the goal and the value of information collection. This cost function here takes the form

$$C(n_a) = \alpha_\tau \tau\left(n_a | n_{root}\right) + \tau^*\left(n_a\right) + \alpha_{\mathcal{I}} \mathcal{I}_a, \tag{B.14}$$

where $\tau(n_a|n_{root})$ is the simulated time to travel from the root node n_{root} to node n_a, $\tau^*(n_a)$ is the lower-bound cost-to-go (e.g. Euclidean or Dubins length divided by average speed) from n_a to the goal, and $\mathcal{I}_a$ is the information-related cost component. The weights α_τ and $\alpha_{\mathcal{I}}$ can be adjusted to reflect the relative importance of information gathering and of following minimal-time paths to the goal. To ensure all recent measurements are taken into account, the latest target estimates are measured at the beginning of each execution loop (line 4), which are then used to update the FIM of each node in the tree (line 9). Though this FIM update is performed on the entire tree on each pass, this is a computationally efficient operation compared to other aspects of the algorithm, such as constraint evaluation.

Of particular note with this cost function is that it can be shown to result in "smooth" mission-level behaviors, in the sense that negligible churning between information collection and goal directedness exists. Rather, the planner is always conscious of the inherent tradeoff and will generate behaviors that, for example, conclude missions by maneuvering to collect information while remaining relatively close to the goal. It should also be noted as a limitation of IRRT, and RRTs in general, that mission-critical requirements like maximum allowable duration and/or minimum required information collection are not well handled; it is difficult enough to find, let alone guarantee that one could find, a feasible solution to such requirements in finite time. Despite this, IRRT has been shown through simulations to perform well empirically under a number of previously prohibitive general constraints. Furthermore, recent flight results have demonstrated the viability of the IRRT approach, incorporating multiple vehicles, complex uncertainty models and sensors in the loop[133].

B.3.3 Recursive Bayesian Hybrid Data Fusion

This section describes the components of the "State Estimation" block in Figure B-2, which combines observations made by human and robot agents to update the Gaussian Mixture target PDFs used by the CBBA task allocation algorithm and IRRT path planning algo-

rithm for each agent. The proposed sensor fusion process is centralized and leads to recursive Bayesian GM updates. Decentralized recursive Bayesian fusion with GMs remains a challenging problem [109] and will be addressed in future work.

Overview of Gaussian Mixture Fusion Updates and Measurement Models

Let $\mathcal{Z}_k = \{\mathbf{Z}_1^{total}, \ldots, \mathbf{Z}_k^{total}\}$ be the set of all available observations up to time k from human and robot agents, where $\mathbf{Z}_k^{total} = \{\mathbf{Z}_k^r, \mathbf{Z}_k^h\}$ is the set of observations $\mathbf{Z}_k^r$ from robot agents at time k and the set of observations $\mathbf{Z}_k^h$ from human agents at time k. For N_r robot agents and N_h human agents, $\mathbf{Z}_k^r$ contains $N_r^k \leq N_r$ measurements $\mathbf{z}_k^{r,j} \in \mathbb{R}^{n_r \times 1}$ for $j \in \{1, \ldots, N_r\}$ and $\mathbf{Z}_k^h$ contains $N_h^k \leq N_h$ measurements $\mathbf{z}_k^{h,j} \in \mathbb{R}^{n_h \times 1}$ for $j \in \{1, \ldots, N_h\}$, where n_r and n_h are the fixed sizes of the robot and human measurement vectors, respectively. The observations $\mathbf{z}_k^{r,j}$ and $\mathbf{z}_k^{h,j}$ are generally non-linear and described stochastically by non-Gaussian likelihood functions $p(\mathbf{z}_k^{r,j}|\mathbf{x}_i)$ and $p(\mathbf{z}_k^{h,j}|\mathbf{x}_i)$, where it is assumed that reliable agent state estimates and environment maps are available so that only $\mathbf{x}_i$ is uncertain. A more general treatment of the fusion problem that includes uncertain target dynamics, agent states and environment maps is also possible, but is omitted here for brevity.

If $\mathbf{Z}_k^r$ and $\mathbf{Z}_k^h$ are conditionally independent given $\mathbf{x}_i$, the Bayesian posterior (Equation (B.5)) can be recursively computed as

$$p(\mathbf{x}_i|\mathcal{Z}_k) = \frac{1}{K} p(\mathbf{x}_i|\mathcal{Z}_{k-1}) p(\mathbf{Z}_k^{total}|\mathbf{x}_i) = \frac{1}{K} p(\mathbf{x}_i|\mathcal{Z}_{k-1}) p(\mathbf{Z}_k^r|\mathbf{x}_i) p(\mathbf{Z}_k^h|\mathbf{x}_i), \tag{B.15}$$

where

$$K = \int p(\mathbf{x}_i|\mathcal{Z}_{k-1}) p(\mathbf{Z}_k^r|\mathbf{x}_i) p(\mathbf{Z}_k^h|\mathbf{x}_i) d\mathbf{x}_i \tag{B.16}$$

is a normalization constant, and

$$\begin{aligned} p(\mathbf{Z}_k^r|\mathbf{x}_i) &= \prod_{\mathbf{z}_k^{r,j} \in \mathbf{Z}_k^r} p(\mathbf{z}_k^{r,j}|\mathbf{x}_i), \\ p(\mathbf{Z}_k^h|\mathbf{x}_i) &= \prod_{\mathbf{z}_k^{h,j} \in \mathbf{Z}_k^h} p(\mathbf{z}_k^{h,j}|\mathbf{x}_i), \\ p(\mathbf{x}_i|\mathcal{Z}_0) &= p(\mathbf{x}_i). \end{aligned} \tag{B.17}$$

Since $p(\mathbf{x}_i|\mathcal{Z}_k) = p(\mathbf{x}_i|\mathcal{Z}_{k-1}, \mathbf{Z}_k^r, \mathbf{Z}_k^h)$, Equation (B.15) is factored into sequential Bayesian

updates for $\mathbf{Z}_k^r$ and $\mathbf{Z}_k^h$,

$$p(\mathbf{x}_i|\mathcal{Z}_{k-1}, \mathbf{Z}_k^r) = \frac{1}{K_r} p(\mathbf{x}_i|\mathcal{Z}_{k-1}) p(\mathbf{Z}_k^r|\mathbf{x}_i) \tag{B.18}$$

$$p(\mathbf{x}_i|\mathcal{Z}_{k-1}, \mathbf{Z}_k^r, \mathbf{Z}_k^h) = \frac{1}{K_h} p(\mathbf{x}_i|\mathcal{Z}_{k-1}, \mathbf{Z}_k^r) p(\mathbf{Z}_k^h|\mathbf{x}_i), \tag{B.19}$$

where

$$K_r = \int p(\mathbf{x}_i|\mathcal{Z}_{k-1}) p(\mathbf{Z}_k^r|\mathbf{x}_i) d\mathbf{x}_i,$$

$$K_h = \int p(\mathbf{x}_i|\mathcal{Z}_{k-1}, \mathbf{Z}_k^r) p(\mathbf{Z}_k^h|\mathbf{x}_i) d\mathbf{x}_i.$$

Finally, (B.18) and (B.19) are evaluated using N_r^k and N_h^k recursive Bayesian updates, respectively,

$$p(\mathbf{x}_i|\mathcal{Z}_{k-1}, ..., \mathbf{z}_k^{r,j}) = \frac{1}{K_k^{r,j}} p(\mathbf{x}_i|\mathcal{Z}_{k-1}, ..., \mathbf{z}_k^{r,j-1}) p(\mathbf{z}_k^{r,j}|\mathbf{x}_i), \tag{B.20}$$

$$p(\mathbf{x}_i|\mathcal{Z}_{k-1}, \mathbf{Z}_k^r, ..., \mathbf{z}_k^{h,j}) = \frac{1}{K_k^{h,j}} p(\mathbf{x}_i|\mathcal{Z}_{k-1}, \mathbf{Z}_k^r, ..., \mathbf{z}_k^{h,j-1}) p(\mathbf{z}_k^{h,j}|\mathbf{x}_i), \tag{B.21}$$

where the indices (r, j) and (h, j) respectively denote the agents with measurements in $\mathbf{Z}_k^r$ and $\mathbf{Z}_k^h$ for $j \in \{1, ..., N_k^r \text{ or } N_k^h\}$, and the constants $K_k^{r,j}$ and $K_k^{h,j}$ are the required normalizing integrals. After (B.18) and (B.19) are evaluated at each time step k for each target i, the updated GM PDFs are fed back to the agents so that they can replan and execute tasks more efficiently via CBBA and IRRT in light of newly acquired information.

$\mathbf{Z}_k^r$ and $\mathbf{Z}_k^h$ are both assumed here to contain discrete multi-category observations with respect to $\mathbf{x}_i$. Prior to actual target acquisition, the discrete target detector outputs for each robot vehicle can be used to update the continuous target PDFs via Equation (B.18), as the probability of detection/no detection function can be treated as the likelihood $p(\mathbf{z}_k^{r,j}|\mathbf{x}_i)$ for the binary observation $\mathbf{z}_k^{r,j} \in \{\text{“no detection”, “detection”}\}$ [47, 77]. Since $\mathbf{z}_k^{r,j}$ depends on the vehicle pose at time k and on $\mathbf{x}_i$, the fusion of “no detection” observations squashes $p(\mathbf{x}_i|\mathcal{Z}_{k-1})$ with vehicle j's “no detection” likelihood at time k, thus shifting probability mass to regions of $\mathcal{S}$ where the target is more likely to be detected.

Similar “GM squashing” updates can also be induced in Equation (B.19) by “soft” human observations $\mathbf{z}_k^{h,j}$ that take the form of ambiguous categorical location descriptions. For instance, a human agent might observe “something is around landmark A”, “something

is behind wall 2", or "nothing is in front of the robot", where the prepositional phrases "around landmark A", "behind wall 2", and "in front of the robot" can be treated as fuzzy categorical labels for coarse range and bearing measurements relative to known locations on a common map. As with probability of detection models, such terms can be modeled probabilistically via likelihood functions that squash the target PDFs towards/away from specified map reference points via Bayes' rule.

The likelihood functions $p(\mathbf{z}_k^{r,j}|\mathbf{x}_i)$ and $p(\mathbf{z}_k^{h,j}|\mathbf{x}_i)$ are modeled here via multimodal softmax (MMS) models, which enable simple piecewise linear representations of "continuous-to-discrete" probability surface mappings [4]. The top left of Figure B-4(b) shows an example 2D MMS model of a triangular probability of detection region for a camera-based target detector mounted to a robot agent facing east. This particular MMS likelihood model has the form

$$p(\mathbf{z}_k^{r,j} = c|\mathbf{x}_i) = \frac{\sum_{s\in\sigma(c)} \exp(w_s^T \xi)}{\sum_{r\in\{\sigma(D)\cup\sigma(ND)\}} \exp\left(w_r^T \xi\right)}, \tag{B.22}$$

where $\xi = [\mathbf{x}_i, 1]^T$, $c \in \{$"no detection" (ND), "detection" (D)$\}$ is the observed category of $\mathbf{z}_k^{r,j}$. The weights w_s in (B.22) are parameters for the two mutually exclusive subcategories, $\sigma(D)$ and $\sigma(ND)$, that geometrically define the possible observation categories of $\mathbf{z}_k^{r,j}$ as a function of ξ, where $s \in \sigma(ND)$ or $s \in \sigma(D)$. The camera detection model in Figure B-4(b) uses 3 subcategories in $\sigma(ND)$ to describe $\mathbf{z}_k^{r,j}$ = "no detection" as being most probable outside of the triangle, and 1 subcategory in $\sigma(D)$ to describe $\mathbf{z}_k^{r,j}$ = "detection" as being most probable inside the triangle. The bottom left of Figure B-4(b) shows an example 2D MMS likelihood model corresponding to a coarse human range observation relative to a robot vehicle, with 3 possible categorical values $\mathbf{z}_k^{h,j} \in \{$"next to", "around", "far from"$\}$. The form of the likelihood function $p(\mathbf{z}_k^{h,j} = c|\mathbf{x}_i)$ for this ternary model is similar to that of the binary camera model in (B.22); here $\sigma($"next to"$)$ contains 1 subcategory (defining the hole of the ring), $\sigma($"around"$)$ contains 8 subcategories (each defining a segment of the octagon ring), and $\sigma($"far from"$)$ contains 8 subcategories (each defining a convex region extending from an outer face of the ring), for a total of 17 weights[5].

Despite their flexibility, the highly nonlinear/non-Gaussian nature of MMS likelihood models means that the exact posteriors on the left-hand sides of Equations (B.20) and (B.21)

[5]See [4] for further details on MMS models.

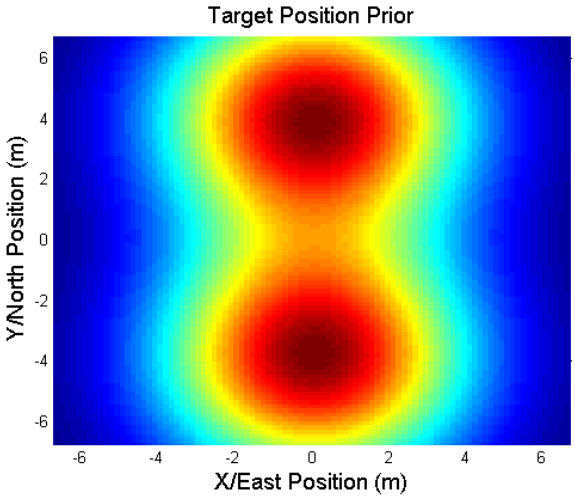
are unfortunately no longer closed-form, since the required integrals for the normalizing constants $K_k^{r,j}$ and $K_k^{h,j}$ cannot be determined analytically. In fact, the resulting hybrid Bayesian updates in (B.20) and (B.21) can only be performed via approximation methods such as discretization or Monte Carlo sampling [131].

Hybrid Bayesian Gaussian Mixture Updates via Monte Carlo Importance Sampling

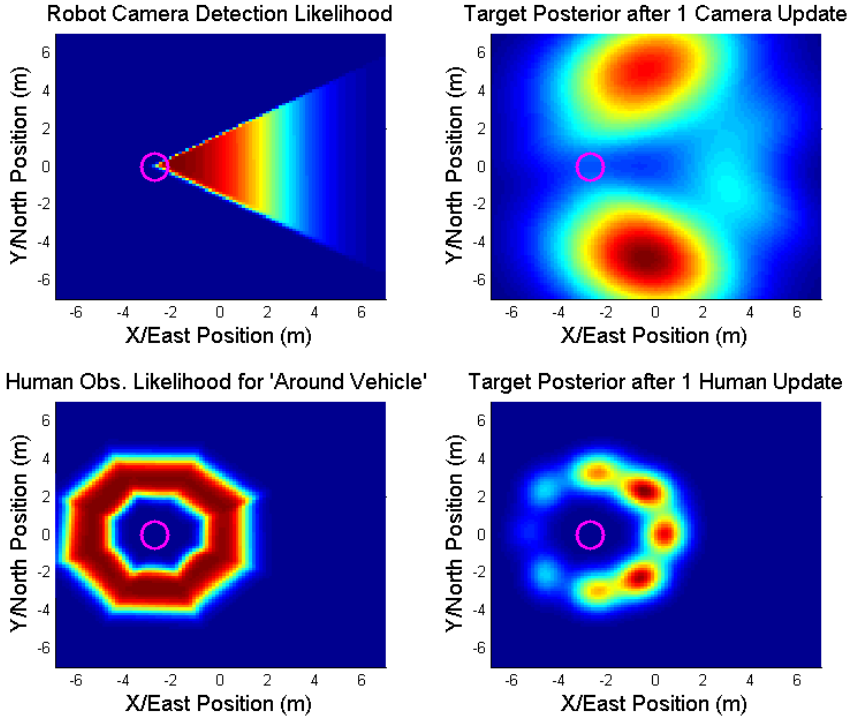
As discussed in [4], fast Monte Carlo importance sampling techniques can be used to obtain accurate GM approximations for the required posteriors on the left-hand sides of Equations (B.20) and (B.21) when the right-hand sides contain GM priors and MMS likelihoods. This leads to a recursive approximate Bayesian fusion strategy in which the priors and posteriors in (B.20) and (B.21) are always represented as GMs, thus ensuring that human and robot agents can incorporate new information from each other in a consistent and compact form.

The basic importance sampling algorithm used here to evaluate Equations (B.20) and (B.21) for non-dynamic GM priors and MMS likelihoods is given in Algorithm 12. Importance sampling approximates expectations under an intractable posterior distribution by using weighted samples drawn from a known "importance" distribution $q(\mathbf{x}_i)$, which ideally has a similar shape to the posterior; this idea underlies the well-known particle filter, which represents the priors and posteriors for non-Gaussian Bayesian filtering via weighted samples at each time step [13]. Algorithm 12 extends this representation by treating the input prior as a full GM and compressing the weighted particles from importance sampling measurement updates into a new GM posterior model. This not only provides a consistent and compact uncertainty representation for task and path planning for all time steps, but also helps avert sample degeneracy problems that can lead to inaccurate/unreliable Bayesian fusion [13].

Since the importance distribution q should be as close as possible to the true posterior for efficient sampling [13], Algorithm 12 also tailors q to the measurement updates in (B.20) and (B.21). For updates via Equation (B.20), q is set to the m^{th} component of the input GM, since its components are typically very close to the true posterior modes when "detection/no detection" observations arrive from robot vehicles at a sufficiently high rate. For updates via Equation (B.21), a variational Bayes algorithm is used to determine a q close to the true posterior, which can shift far away from the modes of the input GM when human agents



(a)



(b)

Figure B-4: (a) Bimodal GM prior with $\mu_1 = [0, 4]^T$, $\mu_2 = [0, -4]^T$, $\Sigma_1 = \Sigma_2 = 5 \cdot \mathrm{I}_2$, and $w_1 = w_2 = 0.5$, (b) *left (top and bottom)*: MMS likelihood models for camera detection probability and human range observation, where probabilities are close to 1 for red and close to 0 for blue; *right (top and bottom)*: corresponding Gaussian mixture posterior approximations for GM prior in (a) (robot vehicle position indicated by magenta circle).

Algorithm 12 Importance Sampling Measurement Update Algorithm

Inputs: Agent type $t \in \{r, h\}$, agent index $j \in \left\{1, ..., N_k^t\right\}$, input GM $p(\mathbf{x}_i|\mathcal{Z}_{k-1}, ..., \mathbf{z}_k^{t,j-1}) = \sum_{m=1}^{M_i} w_{i,m}\, \mathcal{N}(\mu_{i,m}, \Sigma_{i,m})$,
MMS observation likelihood $p(\mathbf{z}_k^{t,j}|\mathbf{x}_i)$, observation $\mathbf{z}_k^{t,j}$ with S_z subcategories in $p(\mathbf{z}_k^{t,j}|\mathbf{x}_i)$
Output: Posterior GM $p(\mathbf{x}_i|\mathcal{Z}_{k-1}, ..., \mathbf{z}_k^{t,j}) = \sum_{h=1}^{S_z \cdot M_i} w_{h,i}\, \mathcal{N}(\mu_{h,i}, \Sigma_{h,i})$
Initialize h $\leftarrow 0$
for Input GM component $m = 1$ to M_i **do**
  for Each subcategory $s \in \sigma(z_k^{t,j})$ **do**
    Set h $\leftarrow$ h + 1
    if $t = r$ for updating via (B.20) **then**
      Set $q(\mathbf{x}_i) = \mathcal{N}(\mu_{i,m}, \Sigma_{i,m})$
    else if $t = h$ for updating via (B.21) **then**
      Set $q(\mathbf{x}_i)$ using variational Bayes method (see [4])
    end if
    1. Draw N samples $\left\{\mathbf{x}_i^1, ..., \mathbf{x}_i^N\right\}$ from $q(\mathbf{x}_i)$
    2. Evaluate importance weights $\left\{\omega^1, ..., \omega^N\right\}$ (see [4])
    3. Estimate the conditional measurement likelihood $l(m, s) = \frac{1}{N} \sum_{n=1}^{N} \omega^n$
    4. Re-normalize importance weights so that $\sum_{n=1}^{N} \omega^n = 1$
    5. Estimate posterior component h mixing weight $w_{h,i}$, mean $\mu_{h,i}$ and covariance $\Sigma_{h,i}$,

$$w_{h,i} = w_{i,m} \cdot l(m, s), \quad \mu_{h,i} = \sum_{n=1}^{N} \omega^n \mathbf{x}_i^n, \quad \Sigma_{h,i} = \sum_{n=1}^{N} \omega^n (\mathbf{x}_i^n \mathbf{x}_i^{nT}) - \mu_{h,i} \mu_{h,i}^T \tag{B.23}$$

  end for
end for
Re-normalize mixing weights so that $\sum_{h=1}^{S_z \cdot M_i} w_{h,i} = 1$

make "surprising" observations (in such cases, severe sample degeneracy could result if q were instead set to the m^{th} component of the input GM). A more detailed description of Algorithm 12, including further details on the selection of q and evaluation of the importance weights ω^n in step 2, can be found in [4].

Figure B-4 illustrates some typical fusion results using Algorithm 12 . Figure B-4(a) shows the 2 component GM prior used in this demonstration with the MMS observation likelihoods shown in Figure B-4(b). The top right of Figure B-4(b) shows the 6 component GM posterior following an update with a robot vehicle "no detection" observation, which pushes probability mass away from the triangular detection region of the robot's camera. The bottom right of Figure B-4(b) shows an 18 component GM posterior following the human observation "target around robot", where almost all of the probability mass from the GM in Figure B-4(a) is forced into the ring-shaped region of the corresponding MMS likelihood model.

Practicalities

Mixture management: If the input GM of Algorithm 12 has M_i components and $p(\mathbf{z}_k^{t,j}|\mathbf{x}_i)$ has S_z relevant subcategories corresponding to $\mathbf{z}_k^{t,j}$, then the output GM will have $M_i \cdot S_z$ components. Thus, GM compression techniques should be used after each measurement update to prevent the number of GM terms in Equations (B.20) and (B.21) from becoming prohibitively large after each time step. As discussed in [4], many GM compression algorithms can be used, although all incur some information loss with respect to the original output GM. Salmond's merging algorithm [193] is used here after each application of Algorithm 12, so that the compressed output GM contains no more than M_{max} mixands that preserve the overall mean and covariance of the uncompressed output GM. While the best value of M_{max} is strongly problem/implementation-dependent, M_{max} must in general be tuned to balance between (i) minimizing the time costs of evaluating and compressing (B.20) and (B.21) for each target (so that agents can replan in a timely manner) and (ii) maintaining the most accurate possible GM approximation of (B.5) for each target (so that agents can use as much information as possible for replanning). To this end, it should also be noted that the fusion updates for multiple independent target GMs can be run in parallel, while the nested **for** loops in Algorithm 12 can be split into $M_i \cdot S_z$ parallel importance sampling updates.

False alarms and data association for ambiguous human observations: It is assumed here that human agents perfectly filter out false target detections from $\mathbf{Z}_k^r$, and that the soft location information in Z_k^h is completely reliable. While it is theoretically possible to extend the proposed fusion process to accommodate false alarms in $\mathbf{Z}_k^r$ and human errors/uncertainties in $\mathbf{Z}_k^h$, these extensions are omitted here for brevity.

Data association issues arise in Equation (B.21) when human observations $\mathbf{z}_k^{h,j}$ are not target specific (e.g. if a human reports the location of "something" or "nothing" in the absence of target ID information). For example, the "positive" observation $\mathbf{z}_k^{h,j}$ ="Something is nearby the robot" could apply to any remaining target, but only truly corresponds to one target. However, the "negative" observation $\mathbf{z}_k^{h,j}$ ="Nothing is nearby the tree" corresponds to all remaining targets. The fusion of such ambiguous measurements can be handled by probabilistic data association techniques [115]. The naive data association method of [4] is used here for the typical case where human observations only describe either "something"

or "nothing" without target ID information.

B.4 Indoor Target Search and Track Experiments

This section describes the experiments conducted at Cornell's Autonomous Systems Lab to validate the proposed planning and fusion methods for an actual cooperative human-robot target search and identification mission. Five static targets (orange traffic cones with numbered labels) were hidden in an obstacle-filled environment and assigned random state priors. A human-robot team consisting of two completely autonomous ground rovers and a human analyst were tasked with detecting, identifying, and localizing all $N = 5$ targets in under 10 minutes. The 5m x 10.5m indoor search area used for the experiment is shown in Figure B-5(a).

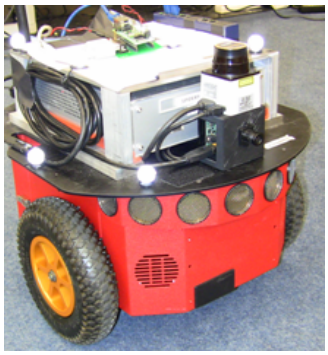
Due to the relatively small scale of the problem, in these experiments, only the robotic agents were involved in the "Task Allocation" and "Path Planning" process described in Figure B-1; the human operator was instead considered a stationary sensor whose sole task was to provide information (i.e. target verification and possibly soft location information) through a desktop computer console to the "State Estimation and Sensor Fusion" block. The targets $i \in \{1, \ldots, 5\}$ were placed at fixed locations $\mathbf{x}_i^{true}$ throughout a field featuring four movable wall obstacles measuring between 1m and 1.5m, which obstructed the human operator's direct line-of-sight to some of the targets when seated at the console. The operator had access to the live video streams from the vehicle cameras, displayed at the console, to assist in the classification process. The operator could also send information to the robots via the Human-Robot Interface (HRI) to improve their autonomous search.

B.4.1 Experimental Setup

Hardware

The robot platform used for these experiments was the Pioneer 3DX with the following mounted devices: Mini ATX based computer with a 2.00 GHz Intel® Core™ 2 processor, 2 GB of RAM and WiFi networking to control the robot, a Unibrain Fire-I OEM Board camera, and a Hokuyo URG-04 LX LIDAR sensor (Figure B-5(b)). A computer with a 2.66 GHz Intel® Core™ 2 Duo processor and 2 GB of RAM performed the State Estimation and Human-Robot Interface (HRI) tasks. An additional computer with similar hardware

(a) Search area used for experiment



(b) Pioneer 3DX rover equipped with computer, LIDAR, and camera

Figure B-5: Real-time search and track experiments for human-robot teams performed at Cornell's Autonomous Systems Lab

executed task allocation in a simulated decentralized environment. A computer with two 2.66 GHz Intel® Xeon® processors and 4 GB of RAM implemented high-level path planning for the experimental trials. A Vicon motion-tracking system performed robot localization and pose calculations.

Messaging, Local Controller, and Target Detection Algorithms

The Pioneers sent three types of messages: images, detected targets, and robot data. Each robot data message was sent at 20 Hz and contained the robot's name, ID number, pose, local timestamp, most recent LIDAR scan, current path, current trajectories, and path completion status. This data was processed for high-level task and path planning, as well as low-level local robot control and target detection.

Local Pioneer robot controllers enabled the vehicles to autonomously follow the waypoint trajectories created by the path planners while avoiding dynamic collisions. The Pioneer controllers were provided with a map of the search environment in order to plan around known obstacles as well as state data for any dynamic obstacles. Local path following used a D* planning algorithm to find an optimal path and a pure pursuit controller to generate velocity commands. The D* planner was required to avoid newly detected obstacles and other robots in the field, as well as to provide a safety measure against possible inconsistencies between Vicon localization and actual field positions. Objects detected by the Pioneer LIDAR units were considered collision hazards if they were within a 0.4m threshold, with a

bearing between 0 and π radians (in front of the vehicle). When a collision was imminent, the robots searched their latest LIDAR scan returns for possible escape directions that avoided these potential collisions. In some cases, a robot could get 'stuck' inside obstacles if they strayed too close to them, which necessitated human intervention to 'rescue' the robot.

The positions of other robots were unknown to the local path planners in this experimental setup, and the local robot controllers were instead responsible for avoiding robot-robot collisions[6]. The state data for each robot was broadcast at 2 Hz so that each local D* planner could model the other robot as a dynamic obstacle. A simple fixed precedence algorithm was used to resolve potential robot-robot collisions if the robots came within a fixed distance of each other. The trajectories sent from the vehicle computer to the Pioneer controller were varied between rates of 4 sec and 10 sec; the effects of these two different rates are examined below.

To detect and localize potential targets, the Pioneer computer used OpenCV "blob" detection algorithms on areas of the camera images that were within thresholds corresponding to the color of a standard traffic cone under the lab's lighting conditions. A cone detection algorithm developed in OpenCV was then employed to provide the HRI with a bounding box of possible cone-shaped blobs in a "potential target detection" message. To simulate realistic mission conditions given the small size of the field, the HRI restricted detection ability to a forward range of 1m by ignoring potential targets with bounding boxes of insufficient height. The HRI associated potentially valid targets with the generated image, and with the LIDAR scan with the closest timestamp to that image. The location of the possible target was then roughly estimated using the LIDAR points with the best match in bearing to the bounding box. If the estimated location was not for an already identified target or a false alarm, this information was prompted to the user, who either identified the specified target or classified it as a false alarm.

Human-Robot Interface

In addition to assisting in target detection, the HRI, shown in Figure B-6, allowed the human operators to send soft sensor information to update the Gaussian mixture model

[6]Note that IRRT has the capability to perform dynamic obstacle/agent avoidance if augmented with the appropriate state data for those dynamic obstacles/agents. This capability was not leveraged in these experiments, however, in the next set of trials this feature will be enabled.

for each target, as described in Section III.B.3.3. The observations from the human agent involved positive/negative information using the previously described "something/nothing" format and the preposition models shown in Section III.B.3.3 to describe target location information relative to field landmarks or the robots themselves via a scroll menu. Using the HRI, the operator also had available the camera views of each Pioneer and a top-down map view of the field, which included all known obstacles, robot locations, identified target/false alarm locations, and the latest combined target PDF. Note that the vehicles were fully autonomous and that the HRI could not directly send any control signals to the robots (except for an emergency stop command). Any operator inputs to the robots were via "soft" observations only, as described in Section III.B.3.3.

B.4.2 Search Performance Metrics and Results

Each target $i \in \{1, \ldots, 5\}$ was assigned a Gaussian prior $p(\mathbf{x}_i) = \mathcal{N}(\mu_i, \Sigma_i)$ with covariance $\Sigma_i = I$ and mean μ_i. The values for μ_i and $\mathbf{x}_i^{true}$ that were used for the experiments were

$$\begin{aligned}
\mu_1 &= [1.50, 1.30]^T, & \mathbf{x}_1^{true} &= [0.103, 1.526]^T \\
\mu_2 &= [4.25, -1.70]^T, & \mathbf{x}_2^{true} &= [2.648, 1.28]^T \\
\mu_3 &= [1.25, 0.55]^T, & \mathbf{x}_3^{true} &= [-0.973, -0.214]^T \\
\mu_4 &= [-1.75, 1.55]^T, & \mathbf{x}_4^{true} &= [2.867, -0.201]^T \\
\mu_5 &= [6.00, 1.05]^T, & \mathbf{x}_5^{true} &= [5.012, -0.679]^T.
\end{aligned}$$

The PDF surface for the combined target prior is shown in Figure B-7, along with the 2D field map, initial search vehicle locations and orientations, and the true target locations and labels. Note that in some cases, $\mathbf{x}_i^{true}$ is sometimes quite far from the specified μ_i prior, which creates a fairly challenging search problem.

Multiple search trials were conducted to compare team search performance using several metrics under different planning and information fusion modalities. The experiments included trials with and without human operator soft inputs and varied task allocation replan rates. Planner performance was evaluated via the following metrics: (1) number of targets detected/identified, (2) individual and combined vehicle distances traveled (not including rotations in place), (3) time required to find the targets, and (4) information acquired throughout the mission. The latter metrics reflect the efficiency of the combined

decentralized planning algorithms in exploring the search space[7]. The information gain at each time-step, following updates from the GM target state estimator, was computed via the Kullback-Leibler divergence (KLD) between the updated combined (undetected) target GM posterior in Equation (B.24) at time-step $k+1$ and the combined target GM posterior from time-step k, given by

$$\mathrm{KL}\left[p(\bar{\mathbf{x}}|\mathcal{Z}_{k+1})||p(\bar{\mathbf{x}}|\mathcal{Z}_k)\right] = \int_{-\infty}^{\infty} p(\bar{\mathbf{x}}|\mathcal{Z}_{k+1}) \ln\left[\frac{p(\bar{\mathbf{x}}|\mathcal{Z}_{k+1})}{p(\bar{\mathbf{x}}|\mathcal{Z}_k)}\right] d\bar{\mathbf{x}}. \tag{B.24}$$

As discussed in [47], the KLD can be thought of as a distance measure between two probability distributions, which quantifies the amount of information acquired about the combined target estimate $\bar{\mathbf{x}}$ from time k to $k+1$. KLDs are nonnegative and are zero only if both PDFs in the integrand are identical, which implies that no new information is acquired. The sum of the KLD over the mission duration reflects the overall average changes in the uncertainty of the undetected target locations during the search mission. The cumulative sum of Equation (B.24) from mission start to end was computed for each trial offline via discretization techniques.

Effect of Human Operator Soft Inputs

To quantify the benefit of the human operator soft inputs to the Bayesian fusion performance, trials were conducted to compared the proposed GM target state estimator which fused soft human information to a baseline GM target state estimator that ignored soft human information. A basic greedy Markov Decision Process (GMDP) path planner served to generate default trajectories. For each robot, the GMDP discretized the combined target PDF for the robot's assigned targets, and then selected the cell with the largest PDF share as the goal location. A path to that location was then planned using value iteration [212], where the robot was allowed to move into adjacent cells by moving either left, right, up, down, or diagonally. For the trials where the soft human observations were ignored, only the probability of no detection information from the robot camera and pose information

[7]Note that mission duration could be less than the allotted 10 minutes either because: all targets were found and identified successfully, or because the trajectory planner or robot controller became "stuck" in an irrecoverable state (e.g. inside an obstacle buffer), in which case the mission was terminated since the robots could not continue autonomously without significant human intervention. Hence, the mission termination condition also served as a loose qualitative measure of planner performance, although it should be emphasized that this is sensitive to the particular tuning and implementations of the local robot controllers and the experimental environment.

were used to update the target GMs. In such cases, the human agent was instructed not to send any observations to the robots, other than responding to target confirmations. For the trials where the human's observations were fused, the human could send soft information observations to the robots at will, as long as the robots were not awaiting target confirmation responses. In all cases, the maximum number of GM components per target was set to 15. The trials were repeated using a 4 sec replan rate for the CBBA task allocation component and a 10 sec replan rate. Table B.1 shows the detected targets along with the time it took the team to find them in each of the trials.

Table B.1: Results for Human Operator Soft Input Experiments: Targets Detected and Time to Detect Targets (in order of acquisition)

Case	Targets, no Human	Time (s), no Human	Targets, with Human	Time (s), with Human
4 sec CBBA	2,5,4,1	98,246,496,543	4,3,2,1,5	25,199,241,286,336
10 sec CBBA	1,4,5,3	65,209,262,427	2,4,3,1,5	49,60,79,347,365

Figure B-8 compares the mission performance for the different trials, showing the vehicle distances traveled (Figure B-8(a)) and the mission durations (Figure B-8(b)) for each case. These results highlight the benefit of the human operator soft inputs, showing that by fusing in the information provided by the operator, the robotic team is able to locate and identify the targets more quickly and efficiently than without the operator inputs.

Information-Based Search and Track

Next, the IRRT trajectory planning approach described in Section III.B.3.2 was implemented and used to generate information-based paths for the robotic agents. Multiple trials were conducted for different task allocation replan rates (4 sec vs. 10 sec), and for cases with and without the human operator soft inputs described above. Table B.2 presents the results for these trials, showing the targets acquired throughout the mission along with the time it took the team to located and identify them. Figure B-9 shows the mission durations and vehicle distances traveled for the different trials, and Figure B-10 shows the information acquired by the team throughout the mission for the different experiments. There are some interesting observations that can be made from these results. Firstly, as shown in Figure B-10, the trials incorporating human soft inputs achieved a higher information content than

the trials without human inputs for both the 10 sec and 4 sec replan cases. In fact, in the 4 sec replan case the autonomous team found only 2 out of the 5 targets, but using human inputs it was able to find 4 (see Table B.2). A second observation is that the information acquired using a 10 sec replan rate was consistently higher than that obtained using a 4 sec replan rate (for both with and without human inputs). This is due to a tradeoff between replanning often to include the latest target estimate information vs. allowing IRRT enough time to generate a quality plan before using it. Finally, it is worth noting that the impact of human inputs and replan rates on vehicle distances and total mission completion times was inconclusive for this set of experiments. This is partly due to the fact that vehicle distances and mission times were affected by the collision avoidance software that would sometimes stop or reroute vehicles before returning them to the trajectories planned by IRRT. The next section provides a more detailed discussion on the different elements that impacted the results, especially with regards to mission times and vehicle distances.

Table B.2: IRRT Targets Detected and Time to Detect Targets (in order of acquisition)

Case	Targets, no Human	Time (s), no Human	Targets, with Human	Time (s), with Human
4 sec CBBA	5,3	148,287	3,4,1,5	77,176,178,282
10 sec CBBA	5,4,2,1	27,85,121,339	4,2,1,5	30,144,222,421

B.4.3 Discussion

The hardware implementation results for the multi-target search and identification mission provide several interesting and useful insights about the proposed information-rich planning and hybrid fusion framework for human-robot teams. This section describes the benefits of the proposed architecture as well as lessons learned and suggestions for future work.

Effects of soft human inputs on Bayesian fusion performance

The performance metrics above show that the fusion of human information generally improved target search efficiency; in particular, the rates of average information gain with respect to the undetected target PDF without human inputs were smaller than the rates of average information gain with human input. This shows that although human agents

are rate limited and less precise than conventional robot sensors, proper Bayesian fusion of soft categorical human data can lead to significant improvements in team performance for information-based tasks. Soft categorical human inputs are especially interesting to consider for data fusion in ISR-type missions, due to their natural interpretability to human operators and their high degree of flexibility. The GM-based data fusion framework presented here for soft human inputs can also readily accommodate existing Bayesian fusion methods for conventional sensors (e.g. such as those based on Kalman filtering).

However, care must always be taken in practice to properly characterize the context of soft human inputs before fusion takes place. For example, the meaning (and hence the likelihood functions) of 'nearby' is quite different in the statements 'the car is nearby the tree' and 'Newark is nearby New York City'. In the experiments presented here, this issue is resolved through the use of a simple messaging protocol and a limited but unambiguous set of contextual cues (i.e. the 'wall' and 'robot' location reference points) that can be passed to an interpreter, which is constructed offline. More sophisticated messaging protocols or interfaces (e.g. natural language) could also be implemented with additional effort for other applications, as long as sufficient training data is available for properly translating the desired soft categories into meaningful probabilistic likelihood functions, as described in Section B.3.

Interestingly, while human sensor inputs are clearly advantageous from an information fusion perspective, they remain challenging to directly accommodate and account for within information-based planning algorithms, since humans are not as predictable or reliable as conventional sensors such as cameras and LIDAR. In particular, the highly intermittent and nonlinear/non-Gaussian nature of soft categorical human inputs makes it difficult to predict the expected amount of information to be contributed by human sensor agents over a finite horizon. As a result, information-based planners must be tuned carefully to the styles of different human operators in order to ensure good team performance. Future work will explore different strategies for coping with these issues, such as explicitly polling the human operator for inputs to improve predictability and calculation of information gain bounds with respect to intermittent human observations.

Analysis of information-based search and track

While the experiments using the proposed IRRT-CBBA fusion architecture illustrated the viability of a unified information-based planning framework, there were several interesting issues observed and lessons learned. Firstly, the experiments using a 4 sec task allocation replan rate vs. a 10 sec replan rate highlighted a tradeoff between plan relevance and plan quality. The quality of the plans generated by IRRT improves the longer the algorithm is allowed to run, thus the 10 sec replan rate created more informative trajectories which enabled the team to acquire larger amount of information throughout the mission (see Figure B-10). On the other hand, a faster replan rate ensures that the current plan remains relevant in light of changing information, such as updated GM estimates and critical information provided by human operators. For this particular mission, a 10 sec task allocation replan rate proved more efficient than a 4 sec rate, but in a more dynamic environment it is likely that the relevance of the plan will become more important than the quality, thus favoring faster replan rates. This tradeoff is problem and implementation dependent, and should be carefully considered when implementing these system to achieve the best performance.

Secondly, the complications associated with performing actual hardware experiments impacted the performance of the system resulting in discrepancies between expected and actual performance. For example, the information-based trajectories planned by the IRRT component were not being executed exactly by the rovers during the experiments for a variety of reasons such as collision avoidance, delays, and modeling inaccuracies. The low-level dynamic collision avoidance software often changed the shape of the trajectory, creating detours so that the vehicles would not hit each other. These detours often resulted in larger vehicle distances and longer mission completion times. Although the IRRT algorithm presented in [133] accounts for dynamic collision avoidance, the distributed hardware nature of the experiment did not allow us to take advantage of this feature, however, this capability will be included in future iterations of these experiments. The other reason mentioned above involved delays between planning the trajectory and communicating it to the vehicles. Although minor, these delays impacted the trajectory following capability and the next iteration of experiments will attempt to minimize these delays as much as possible. The third reason was due to mismatches between the IRRT's model of the vehicle dynamics and sensor model, and those actually used by the rovers during execution (such as many of the

D* effects). IRRT specifies exact trajectories that include position, orientation and sensor location, but the low-level path following software consisted only of waypoint following, causing discrepancies between the planned and actual sensor location/orientation required for obtaining accurate measurements (although the vehicles often got really close to taking a proper measurement, they would pass by without looking directly at the target, and therefore there would be no reward for executing the almost perfect trajectory). In addition, the actual measurement procedure was slightly different than that modeled in the IRRT framework, causing mismatches in predicted and actual performance. In particular, the vehicle sensor limitations in the vision processing software were more conservative than the sensor models used by IRRT. Thus, while the trajectories generally exhibited good behavior, the "reward" in terms of valid observations was often insufficient. Furthermore, there was significant sensitivity in the actual measurement procedure making it difficult to obtain a good measurement. For example, the target cones had to be centered at a certain height and distance for the vision processing software to accept them, and white labels used to mark the targets were interfering with the vision software's classifiers, therefore, even when the vehicles were looking directly at the targets, a proper measurement was not received. These real-world considerations are hard to model within the IRRT framework, and future work should consider ways to incorporate robustness into both the planning and image processing components.

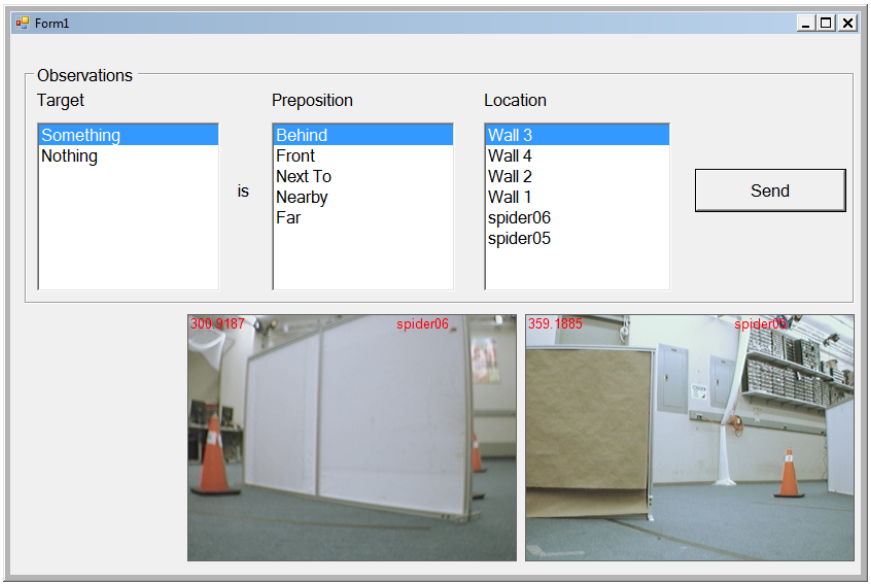
Lastly, it should also be noted that, out of all these trials, only the two GMDP trials with human input terminated in under 10 minutes because all targets were successfully detected and identified. All other trials ended either because the 10 minute time limit was reached (GMDP, no human, 4 sec CBBA replan rate) or because the robots determined that they were irreversibly "trapped" by obstacles (often by moving too close to previously detected targets or walls), thus requiring human intervention to break free (all other trials). To ensure some consistency for meaningful comparisons in light of these difficulties, these latter trials were only accepted after at least 5 minutes of the search had elapsed, otherwise the trial was discarded and restarted. As mentioned before, early termination due to motion infeasibility was usually caused by discrepancies in the low level controllers, Vicon state estimation, and/or environmental setup, which are issues that will be addressed in the next iteration of trials. In spite of all the real-world considerations that arise when dealing with actual hardware, note that at least 4 out of the 5 targets were successfully detected and

identified in all but one of the eight total trials, illustrating the potential benefits of this information-based planning and fusion architecture for search and track missions.

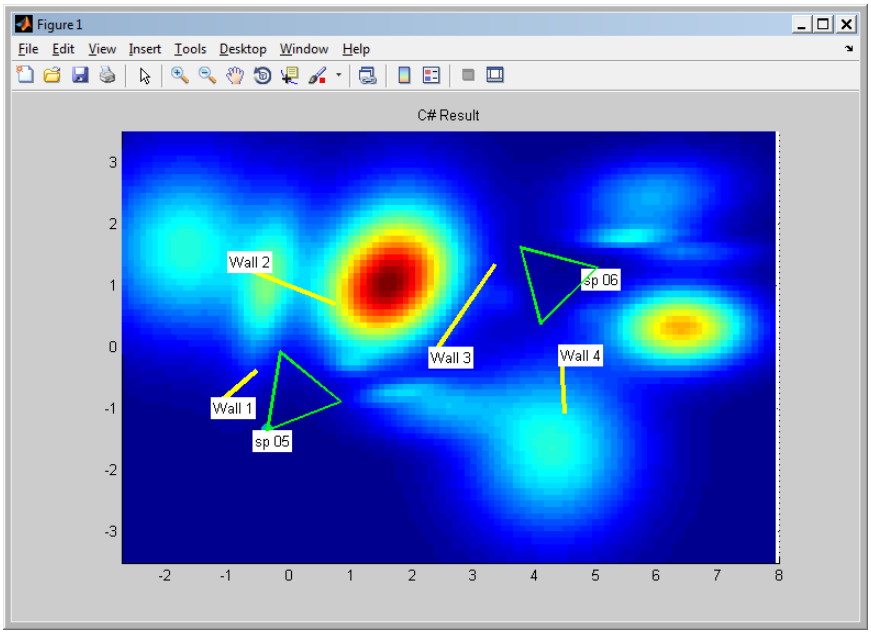
B.5 Conclusions and Ongoing Work

Motivated by the need for scalable and flexible information-based planning algorithms and robust fusion methods to exploit heterogeneous information sources in large-scale semi-autonomous systems, this paper introduces a new planning and estimation framework for optimizing information collection in cooperative human-robot missions. To this end, a unified approach to high-level distributed task planning, information-based path planning, and hybrid Bayesian information fusion using Gaussian mixtures is presented and validated in a real hardware experiment involving multi-target search with a cooperative human robot team. The results illustrate the benefits of including information acquisition as a goal at every level of planning, as well as showing that by including human operator soft inputs into the Bayesian fusion framework the performance and efficiency of the autonomous search team can be greatly improved.

Future work will consider extending the proposed planning and fusion framework in several ways to accommodate other realistic estimation problems for cooperative human-robot search missions. These extensions include (but are not limited to): dynamic target tracking with continuous sensor information fusion and multiple model uncertainties; environmental map uncertainties to accommodate simultaneous localization and mapping (SLAM) [212]; 3D target dynamics and sensor model updates (e.g. using UAV sensor platforms); decentralized fusion with Gaussian mixtures; and improved false alarm modeling and data association techniques for soft information fusion. Future work will also consider task-planning extensions for human sensor agents and for human-operated mobile agents, as well as task and path planning for mobile human agents. Since CBBA is well-suited to handling heterogeneous agents that can perform a wide variety of tasks, extensions to incorporate realistic secondary mission objectives such as refueling and automation failure handling (e.g. assignment of a human agent to tele-operate a 'trapped' robot to rescue it) will also be studied.



(a) Main operator user interface window, with menus for soft inputs and windows for camera feeds



(b) Overhead field map with the combined target GM PDF, walls, and robot locations

Figure B-6: Screenshots from the HRI console available to the human operator

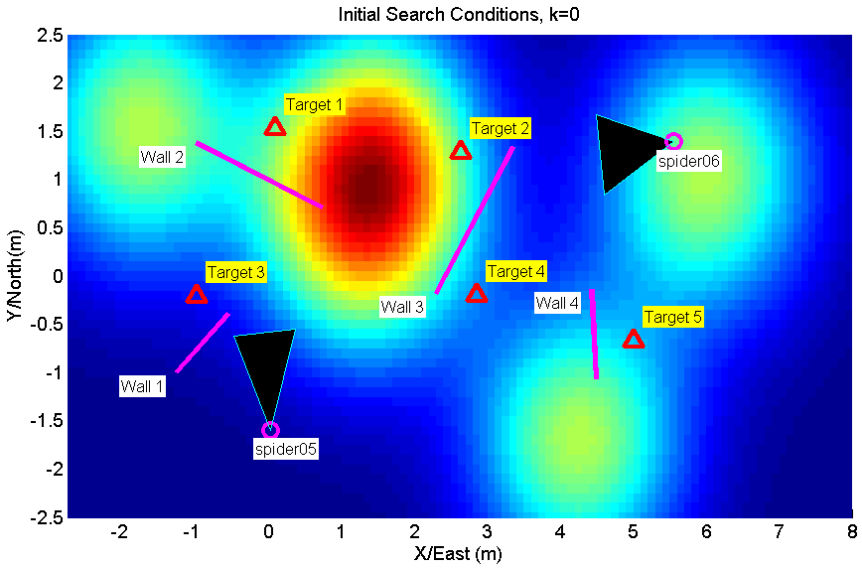
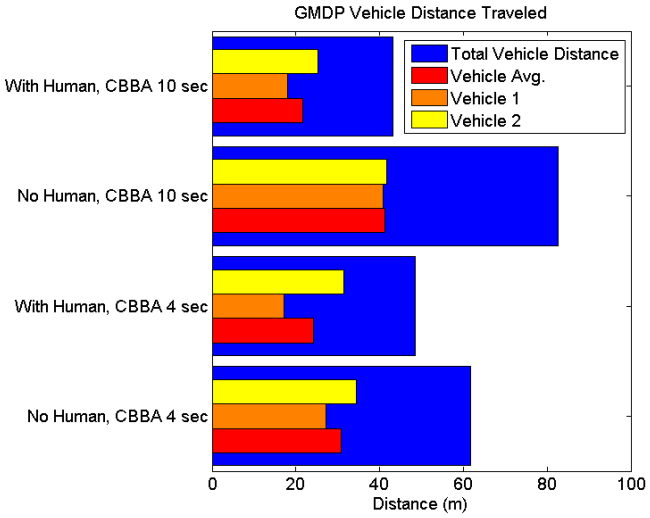
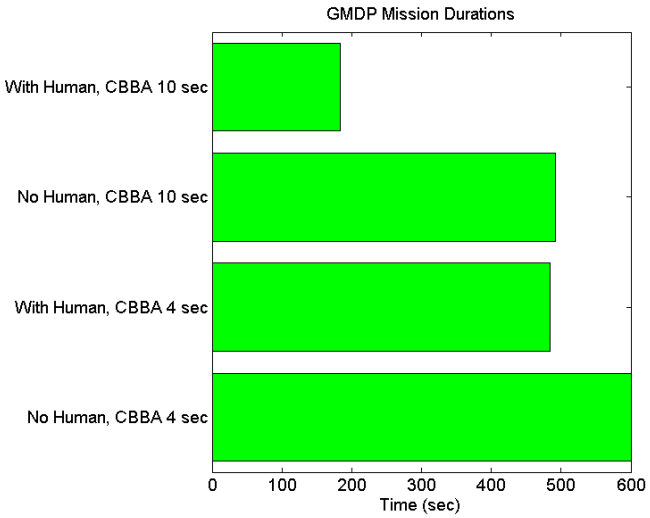


Figure B-7: Field map showing walls (magenta lines), true target locations (red triangles), initial target prior for combined target GM, and initial vehicle locations (circles) with camera detection field of view (black triangles).

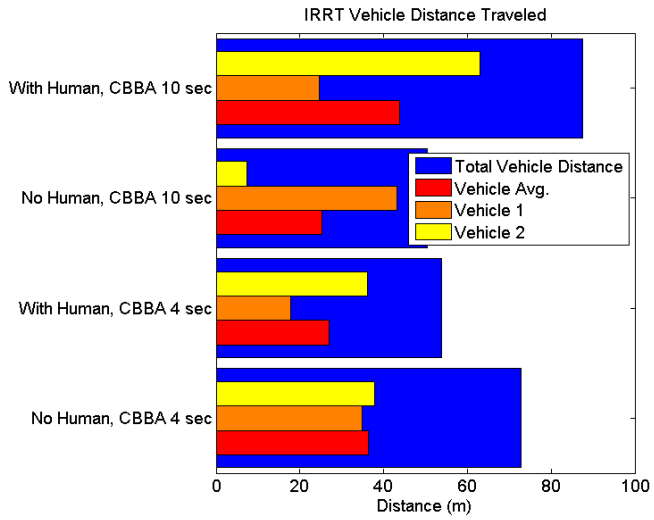


(a) Individual and combined vehicle distances traveled during each trial

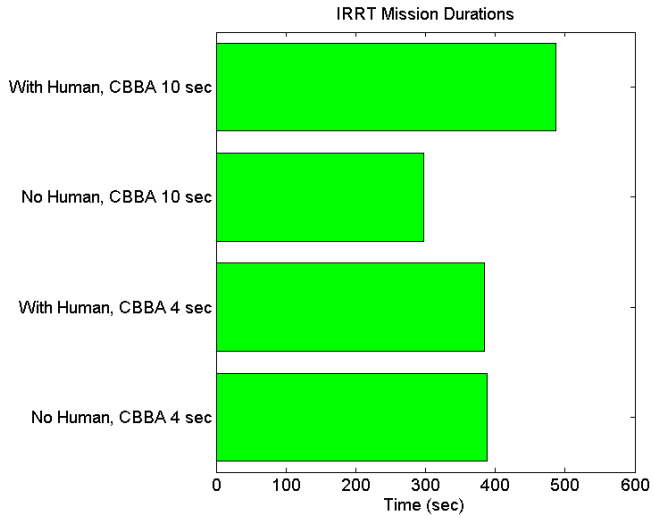


(b) Total mission duration for each trial

Figure B-8: Results for Human Operator Soft Input Experiments: Comparison of mission durations and distances traveled with and without human operator soft inputs.



(a) Individual and combined vehicle distances traveled during each trial



(b) Total mission duration for each trial

Figure B-9: Results for Information-Based Search and Track Experiments: Comparison of mission durations and distances traveled

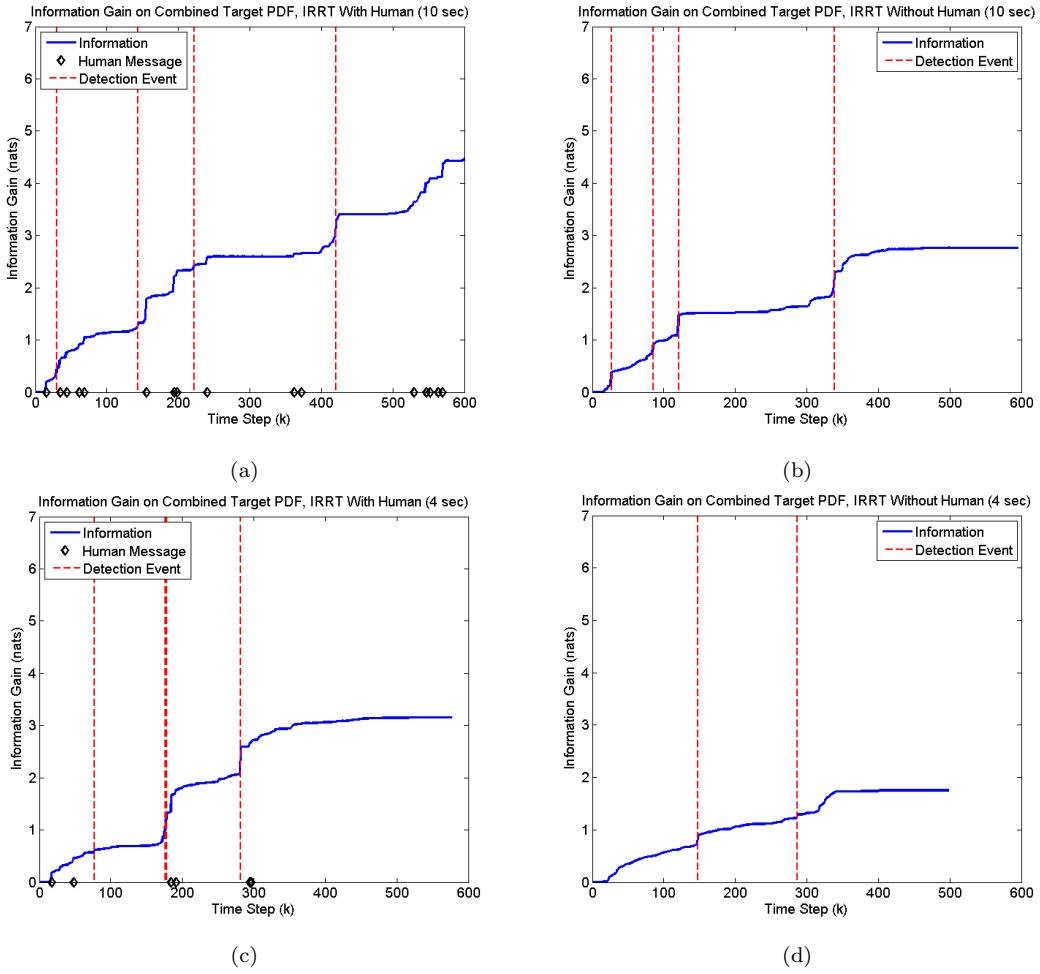


Figure B-10: Results for Information-Based Search and Track Experiments: KLD information gain for the combined undetected target PDF. Plots show information acquired for (a) 10 sec CBBA replan with human inputs, (b) 10 sec CBBA replan without human inputs, (c) 4 sec CBBA replan with human inputs, and (d) 4 sec CBBA replan without human inputs. Red lines indicate target detection/ID events and black diamonds denote instances where a soft human observation message is fused.

Bibliography

[1] FY2009-2034: Unmanned systems integrated roadmap. Technical report, Office of the Secretary of Defense, USA, April 2009.

[2] Technology horizons: A vision for air force science and technology during 2010-2030. Technical report, Office of the Chief Scientist of the U.S. Air Force (AF/ST), May 2010.

[3] A. Ahmed, A. Patel, T. Brown, M. Ham, M. Jang, and G. Agha. Task assignment for a physical agent team via a dynamic forward/reverse auction mechanism. In *International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 2005.

[4] N. Ahmed, E. Sample, K. Ho, T. Hoossainy, and M. Campbell. Categorical soft data fusion via variational bayesian importance sampling with applications to cooperative search. In *American Control Conference (ACC)*, pages 1268–1273. IEEE, 2011.

[5] B. Alidaee, H. Wang, and F. Landram. A note on integer programming formulations of the real-time optimal scheduling and flight path selection of UAVs. *Control Systems Technology, IEEE Transactions on*, 17(4):839–843, 2009.

[6] B. Alidaee, H. Wang, and F. Landram. On the flexible demand assignment problems: Case of unmanned aerial vehicles. *Automation Science and Engineering, IEEE Transactions on*, (99):1–1, 2011.

[7] M. Alighanbari and J. P. How. Decentralized task assignment for unmanned aerial vehicles. In *IEEE Conference on Decision and Control (CDC)*, pages 5668–5673, 12–15 Dec. 2005.

[8] M. Alighanbari and J. P. How. An Unbiased Kalman Consensus Algorithm. *AIAA Journal of Aerospace Computing, Information, and Communication*, 5(9):298–311, Sept 2008.

[9] M. Alighanbari and J. P. How. A robust approach to the UAV task assignment problem. *International Journal of Robust and Nonlinear Control*, 18(2):118–134, January 2008.

[10] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, 2000.

[11] C. Andrieu, N. De Freitas, A. Doucet, and M.I. Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1):5–43, 2003.

[12] G. Arslan, J. R. Marden, and J. S. Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129:584, 2007.

[13] M.S. Arulumpalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. 50(2):174–188, 2002.

[14] M. L. Atkinson. Results analysis of using free market auctions to distribute control of UAVs. In *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, 2004.

[15] H. Balakrishnan and B. Chandran. Algorithms for Scheduling Runway Operations under Constrained Position Shifting. *Operations Research*, 58(6), 2010.

[16] A. G. Banerjee, M. Ono, N. Roy, and B. C. Williams. Regression-based LP solver for chance-constrained finite horizon optimal control with nonconvex constraints. In *Proceedings of the American Control Conference*, San Francisco, CA, 2011.

[17] C. Barnhart and A. Cohn. Commissioned Paper Airline Schedule Planning: Accomplishments and Opportunities. *Manufacturing & service operations management*, 6(1):3–22, 2004.

[18] C. Barnhart and Massachusetts Institute of Technology. *Planning and Control of Transportation Systems: Robust Airlines Planning*. New England University Transportation Center, Massachusetts Institute of Technology, 2000.

[19] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson. Coordinated Target Assignment and Intercept for Unmanned Air Vehicles. *IEEE Transactions on Robotics and Automation*, 18:911–922, 2002.

[20] R.W. Beard and V. Stepanyan. Information consensus in distributed multiple vehicle coordinated control. In *IEEE Conference on Decision and Control (CDC)*, volume 2, pages 2029–2034, Dec. 2003.

[21] R. Becker. Solving transition independent decentralized markov decision processes. *Computer Science Department Faculty Publication Series*, page 208, 2004.

[22] J. Bellingham, A. Richards, and J. P. How. Receding horizon control of autonomous aerial vehicles. In *American Control Conference (ACC)*, volume 5, pages 3741–3746, 2002.

[23] Richard Bellman. *Dynamic Programming*. Dover Publications, March 2003.

[24] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, pages 769–805, 1998.

[25] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–14, 1999.

[26] D.S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, pages 819–840, 2002.

[27] D. P. Bertsekas. The auction algorithm for assignment and other network flow problems. Technical report, MIT, 1989.

[28] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I-II, 3rd Ed.* Athena Scientific, Belmont, MA, 2007.

[29] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.

[30] D.P. Bertsekas and D.A. Castanon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5(1):89–108, 1999.

[31] D. Bertsimas. *Probabilistic combinatorial optimization problems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1988.

[32] D. Bertsimas and D.B. Brown. Constructing uncertainty sets for robust linear optimization. *Operations research*, 57(6):1483–1495, 2009.

[33] D. Bertsimas, D.B. Brown, and C. Caramanis. Theory and applications of robust optimization. *Arxiv preprint arXiv:1010.5445*, 2010.

[34] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1):49–71, 2003.

[35] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[36] D. Bertsimas and R. Weismantel. *Optimization over integers*. Dynamic Ideas Belmont, MA, 2005.

[37] L. F. Bertuccelli, H.-L. Choi, P. Cho, and J. P. How. Real-time Multi-UAV Task Assignment in Dynamic and Uncertain Environments. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2009 (AIAA 2009-5776).

[38] L. F. Bertuccelli and J. P. How. Uav search for dynamic targets with uncertain motion models. In *IEEE Conference on Decision and Control (CDC)*, pages 5941–5946, 13-15 Dec. 2006.

[39] Luca Bertuccelli and Jonathan How. Active exploration in robust unmanned vehicle task assignment. *Journal of Aerospace Computing, Information, and Communication*, 8:250–268, 2011.

[40] Luca F. Bertuccelli. Robust planning for heterogeneous UAVs in uncertain environments. Master's thesis, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA, June 2004.

[41] B. Bethke, J. P. How, and J. Vian. Group health management of UAV teams with applications to persistent surveillance. In *American Control Conference (ACC)*, pages 3145–3150, Seattle, WA, 11-13 June 2008.

[42] Brett M. Bethke. *Kernel-Based Approximate Dynamic Programming Using Bellman Residual Elimination*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, February 2010.

[43] J.R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Verlag, 1997.

[44] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[45] L. Blackmore and M. Ono. Convex chance constrained predictive control without sampling. *AIAA Proceedings.[np]. 10-13 Aug*, 2009.

[46] Vincent D. Blondel, Julien M. Hendrickx, Alex Olshevsky, and John N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proceedings of the IEEE Conference on Decision and Control*, 2005.

[47] Frédéric Bourgault. *Decentralized Control in a Bayesian World*. PhD thesis, University of Sydney, 2005.

[48] S. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning. *Journal of Machine Learning Research (JMLR)*, 22:33–57, 1996.

[49] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Boca Raton, Florida, 2010.

[50] J. Capitán, M.T.J. Spaan, L. Merino, and A. Ollero. Decentralized multi-robot cooperation with auctioned pomdps. In *Sixth Annual Workshop on Multiagent Sequential Decision Making in Uncertain Domains (MSDM-2011)*, page 24, 2011.

[51] D. A. Castanon and C. Wu. Distributed algorithms for dynamic reassignment. In *IEEE Conference on Decision and Control (CDC)*, volume 1, pages 13–18, 9-12 Dec. 2003.

[52] D.A. Castanon and J.M. Wohletz. Model predictive control for stochastic resource allocation. *Automatic Control, IEEE Transactions on*, 54(8):1739 –1750, aug. 2009.

[53] P. R. Chandler, M. Pachter, D. Swaroop, J. M. Fowler, J. K. Howlett, S. Rasmussen, C. Schumacher, and K. Nygard. Complexity in UAV Cooperative Control. In *American Control Conference (ACC)*, Anchorage AK, May 2002.

[54] A. C. Chapman, R. A. Micillo, R. Kota, and N. R. Jennings. Decentralized Dynamic Task Allocation Using Overlapping Potential Games. *The Computer Journal*, 2010.

[55] A. Charnes and W.W. Cooper. Chance-constrained programming. *Management Science*, 6(1):73–79, 1959.

[56] W. Chen, M. Sim, J. Sun, and C.P. Teo. From cvar to uncertainty set: Implications in joint chance constrained optimization. *Operations research*, 58(2):470–485, 2010.

[57] T. Chockalingam and S. Arunkumar. A randomized heuristics for the mapping problem: The genetic approach. *Parallel Computing*, 18(10):1157–1165, 1992.

[58] H.-L. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, August 2009.

[59] R.R. Clewlow, I. Simaiakis, and H. Balakrishnan. Impact of Arrivals on Departure Taxi Operations at Airports. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, August 2010.

[60] T.H. Cormen. *Introduction to algorithms*. The MIT press, 2001.

[61] E. Craparo, J. P. How, and E. Modiano. Throughput optimization in mobile backbone networks. In *SIAM Conference on Optimization*, May 2008.

[62] J.B. Cruz Jr, G. Chen, D. Li, and X. Wang. Particle swarm optimization for resource allocation in uav cooperative control. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–11. Providence, USA, 2004.

[63] J. Curtis and R. Murphey. Simultaneous area search and task assignment for a team of cooperative agents. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2003 (AIAA-2003-5584).

[64] P.T. De Boer, D.P. Kroese, S. Mannor, and R.Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

[65] S. de Vries and R. Vohra. Combinatorial auctions: A survey. *INFORMS Journal of Computing*, 15(3):284–309, 2003.

[66] E. Delage and S. Mannor. Percentile optimization for markov decision processes with parameter uncertainty. *Operations research*, 58(1):203–213, 2010.

[67] P. Dellaportas and G.O. Roberts. Introduction to mcmc. 2001.

[68] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[69] M. Bernardine Dias and Anthony Stentz. A free market architecture for distributed control of a multirobot system. In *6th International Conference on Intelligent Autonomous Systems IAS-6*, pages 115–122, 2000.

[70] C. Dixon and E.W. Frew. Maintaining optimal communication chains in robotic sensor networks using mobility control. *Mobile Networks and Applications*, 14(3):281–291, 2009.

[71] R. Dondo and J. Cerdá. An MILP framework for dynamic vehicle routing problems with time windows. *Latin American Applied Research*, 36(4):255–261, 2006.

[72] Y. Eun and H. Bang. Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithms. *Journal of aircraft*, 46(1):338, 2010.

[73] M. Evans and T. Swartz. Methods for approximating integrals in statistics with special emphasis on bayesian integration problems. *Statistical Science*, 10(3):254–272, 1995.

[74] F.J. Fabozzi, P.N. Kolm, and D. Pachamanova. *Robust portfolio optimization and management*. Wiley, 2007.

[75] Amir Massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized policy iteration. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 441–448. MIT Press, 2008.

[76] J.A. Fax and R.M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, Sept. 2004.

[77] S. Ferrari, G. Foderaro, and A. Tremblay. A probability density function approach to distributed sensors' path planning. In *Proc. of the 2010 Int'l Conf. on Robotics and Automation (ICRA 2010)*, pages 432–439, Anchorage, Alaska.

[78] Ronald A Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London, Series A: Mathematical and Physical Sciences*, 222:309–368, 1922.

[79] E. B. Fox, E. B. Sudderth, and A. S. Willsky. Hierarchical Dirichlet processes for tracking maneuvering targets. In *Proc. International Conference on Information Fusion*, July 2007.

[80] E.B. Fox, D.S. Choi, and A.S. Willsky. Nonparametric bayesian methods for large scale multi-target tracking. In *Fortieth Asilomar Conference on Signals, Systems and Computers (ACSSC '06)*, pages 2009 –2013, Nov 2006.

[81] E.B. Fox, E.B. Sudderth, M.I. Jordan, and A.S. Willsky. Nonparametric Bayesian learning of switching linear dynamical systems. *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[82] C. S. R. Fraser, L. F. Bertuccelli, and J. P. How. Reaching consensus with imprecise probabilities over a network. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Chicago, IL, August 2009 (AIAA-2009-5655).

[83] Cameron S.R. Fraser, Luca F. Bertuccelli, Han-Lim Choi, and Jonathan P. How. A hyperparameter consensus method for agreement under uncertainty. *Automatica*, 48(2):374–380, February 2012.

[84] E. W. Frew and B. Argrow. Embedded reasoning for atmospheric science using unmanned aircraft systems. In *AAAI 2010 Spring Symposium on Embedded Reasoning: Intelligence in Embedded Systems*, Palo Alto, CA, 2010.

[85] D Fudenberg and J Tirole. *Game Theory*. MIT Press, 1991.

[86] A.E. Gelfand. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

[87] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman and Hall, 2nd edition, 2004.

[88] Alborz Geramifard. *Practical Reinforcement Learning Using Representation Learning and Safe Exploration for Large Scale Markov Decision Processes*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, February 2012.

[89] Alborz Geramifard, Finale Doshi, Joshua Redding, Nicholas Roy, and Jonathan How. Online discovery of feature dependencies. In Lise Getoor and Tobias Scheffer, editors, *International Conference on Machine Learning (ICML)*, pages 881–888. ACM, June 2011.

[90] B. Gerkey and M. Mataric. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[91] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, 2004.

[92] F. Glover and R. Marti. Tabu search. *Metaheuristic Procedures for Training Neutral Networks*, pages 53–69, 2006.

[93] C.V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 137–144. ACM, 2003.

[94] C.V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *J. Artif. Intell. Res. (JAIR)*, 22:143–174, 2004.

[95] S. Grime and H.F. Durrant-Whyte. Data fusion in decentralized sensor networks. *Control Engineering Practice*, 2(5):849 – 863, 1994.

[96] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 1523–1530. MIT Press, 2001.

[97] W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97, 1970.

[98] Y. Hatano and M. Mesbahi. Agreement over random networks. *IEEE Transactions on Automatic Control*, 50(11):1867–1872, Nov 2005.

[99] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, 28(2):51–64, April 2008.

[100] M. A. Hsieh, A. Cowley, R. V. Kumar, and C. J. Taylor. Maintaining network connectivity and performance in robot teams. *IEEE Journal*, 25(1-2):111–131, 2008.

[101] A. S. Ibrahim, K.G. Seddik, and K. J. R. Liu. Connectivity-aware network maintenance via relays deployment. *IEEE Transactions on Wireless Communications*, 8(1):356–366, January 2009.

[102] ILOG. Cplex, 2006. http://www.ilog.com/products/cplex/.

[103] A. Jadbabaie, Jie Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.

[104] Nidal Jodeh and Mark Mears. An overview of the cooperative operations in urban terrain (counter) program. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2008.

[105] Luke Johnson. Decentralized Task Allocation for Dynamic Environments. Master's thesis, Massachusetts Institute of Technology, January 2012.

[106] Luke B. Johnson, Han-Lim Choi, Sameera S. Ponda, and Jonathan P. How. Allowing non-submodular score functions in distributed task allocation. In *IEEE Conference on Decision and Control (CDC)*, Dec 2012 (to appear).

[107] Luke B. Johnson, Sameera S. Ponda, Han-Lim Choi, and Jonathan P. How. Asynchronous decentralized task allocation for dynamic environments. In *Proceedings of the AIAA Infotech@Aerospace Conference*, St. Louis, MO, March 2011.

[108] J. Joseph, F. Doshi-Velez, and N. Roy. A Bayesian Nonparametric Approach to Modeling Mobility. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*. AAAI, 2010.

[109] S.J. Julier. An empirical study into the use of chernoff information for robust, distributed fusion of gaussian mixture models. In *FUSION 2006*, pages 1–8, 2006.

[110] Q. Jun, J. Wang, and B. Zheng. A Hybrid Multi-objective Algorithm for Dynamic Vehicle Routing Problems. *Lecture Notes in Computer Science*, 5103:674–681, 2008.

[111] T Kaupp, B. Douillard, F. Ramos, A. Makarenko, and B. Upcroft. Shared environment representation for a human-robot team performing information fusion. *Journal of Field Robotics*, 24(11):911–942, 2007.

[112] Y. Kim, D.W. Gu, and I. Postlethwaite. Real-time optimal mission scheduling and flight path selection. *Automatic Control, IEEE Transactions on*, 52(6):1119–1123, 2007.

[113] E. King, Y. Kuwata, M. Alighanbari, L. Bertuccelli, and J. P. How. Coordination and control experiments on a multi-vehicle testbed. In *American Control Conference (ACC)*, pages 5315–5320, Boston, MA, 30 June-2 July 2004.

[114] E. King, Y. Kuwata, M. Alighanbari, and J. How. Coordination and control experiments for uav teams. *Advances in the Astronautical Sciences*, 118:1–11, 2004.

[115] T. Kirubarajan and Y. Bar-Shalom. Probabilistic data association techniques for target tracking in clutter. *Proc. of the IEEE*, 92(3):536 – 557, 2004.

[116] Andrew N. Kopeikin. Dynamic Mission Planning for Communication Control in Multiple Unmanned Aircraft Teams. Master's thesis, Massachusetts Institute of Technology, June 2012.

[117] Andrew N. Kopeikin, Sameera S. Ponda, Luke B. Johnson, and Jonathan P. How. Multi-UAV Network Control through Dynamic Task Allocation: Ensuring Data-Rate and Bit-Error-Rate Support. In *Wi-UAV 2012, 3rd International Workshop on Wireless Networking and Control for Unmanned Autonomous Vehicles at the IEEE GlobeComm Conference*, Dec 2012 (to appear).

[118] Andrew N. Kopeikin, Sameera S. Ponda, Luke B. Johnson, Olivier Toupet, and Jonathan P. How. Real-Time Dynamic Planning to Maintain Network Connectivity in a Team of Heterogeneous Unmanned Systems. In *Wi-UAV 2011, 2nd International Workshop on Wireless Networking for Unmanned Autonomous Vehicles at the IEEE GlobeComm Conference*, Dec 2011.

[119] J. Kotecha and P.M. Djuric. Gaussian sum particle filtering. *IEEE Trans. on Sig. Proc.*, 51(10):2602–2612, 2003.

[120] P. Krokhmal, R. Murphey, P. Pardalos, S. Uryasev, and G. Zrazhevsky. Robust decision making: Addressing uncertainties in distributions. *Cooperative Control: Models, Applications, and Algorithms*, pages 165–185, 2003.

[121] P. Krokhmal, J. Palmquist, and S. Uryasev. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of Risk*, 4:43–68, 2002.

[122] Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How. Decentralized robust receding horizon control for multi-vehicle guidance. In *American Control Conference (ACC)*, pages 2047–2052, Minneapolis, MN, June 2006.

[123] Y. Kuwata, T. Schouwenaars, A. Richards, and J. P. How. Robust constrained receding horizon control for trajectory planning. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, San Francisco, CA, August 2005 (AIAA-2005-6079).

[124] Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 4:1107–1149, 2003.

[125] S. Lan, J.P. Clarke, and C. Barnhart. Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation Science*, 40(1):15–28, 2006.

[126] G. Laporte and F. Semet. Classical Heuristics for the Capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*. SIAM, Philadelphia, 2002.

[127] S. Leary, M. Deittert, and J. Bookless. Constrained uav mission planning: A comparison of approaches. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2002 –2009, nov. 2011.

[128] H. Lee and H. Balakrishnan. Fuel cost, delay and throughput tradeoffs in runway scheduling. In *American Control Conference, 2008*, pages 2449–2454. IEEE, 2008.

[129] H. Lee, I. Simaiakis, and H. Balakrishnan. A comparison of aircraft trajectory-based and aggregate queue-based control of airport taxi processes. In *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th*, pages 1–B. IEEE, 2010.

[130] T. Lemaire, R. Alami, and S. Lacroix. A Distributed Task Allocation Scheme in Multi-UAV Context. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3622–3627, 2004.

[131] U. Lerner. *Hybrid Bayesian Networks for Reasoning About Complex Systems*. PhD thesis, Stanford University, 2002.

[132] D. Levine, B. Luders, and J. P. How. Information-rich path planning with general constraints using rapidly-exploring random trees. In *AIAA Infotech@Aerospace Conference*, Atlanta, GA, April 2010. (AIAA-2010-3360).

[133] Daniel S. Levine. Information-rich path planning under general constraints using rapidly-exploring random trees. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, June 2010.

[134] M. Lewis, H. Wang, P. Velgapudi, P. Scerri, and K. Sycara. Using humans as sensors in robotic search. In *FUSION 2009*, pages 1249–1256.

[135] J. Lin, A.S. Morse, and B.D.O. Anderson. The multi-agent rendezvous problem. In *IEEE Conference on Decision and Control (CDC)*, volume 2, pages 1508–1513, Dec. 2003.

[136] G.F. List, B. Wood, L.K. Nozick, M.A. Turnquist, D.A. Jones, E.A. Kjeldgaard, and C.R. Lawton. Robust optimization for fleet planning under uncertainty. *Transportation Research Part E: Logistics and Transportation Review*, 39(3):209–227, 2003.

[137] B. Luders, S. Karaman, E. Frazzoli, and J. P. How. Bounds on track error using closed-loop rapidly-exploring random trees. In *American Control Conference (ACC)*, pages 5406–5412, Baltimore, MD, June/July 2010.

[138] Sridhar Mahadevan, Mauro Maggioni, and Carlos Guestrin. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research (JMLR)*, 8:2007, 2006.

[139] Alexei Makarenko and Hugh Durrant-Whyte. Decentralized Bayesian algorithms for active sensor networks. *International Conference on Information Fusion*, 7(4):418 – 433, 2006.

[140] Nguyen Duc Manh, Le Thi Hoai An, and Pham Dinh Tao. A Cross-Entropy Method for Nonlinear UAV Task Assignment Problem. In *IEEE International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, pages 1–5, Nov 2010.

[141] J. R. Marden, G. Arslan, and J. S. Shamma. Cooperative control and potential games. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(6):1393–1407, 2009.

[142] J. R. Marden and A. Wierman. Overcoming Limitations of Game-Theoretic Distributed Control. *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, 2009.

[143] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[144] Lavanya Marla. Robust optimization for network-based resource allocation problems under uncertainty. Master's thesis, Department of Civil and Environmental Engineering and Operations Research Center, MIT, Cambridge, MA, June 2007.

[145] Maja J. Mataric, Gaurav S. Sukhatme, and Esben H. Ostergaard. Multi-robot task allocation in uncertain environments. *Autonomous Robots*, vol. 14, no. 2-3, pp. 255-263, 2003.

[146] I. Maza, F. Caballero, J. Capitán, JR Martínez-de Dios, and A. Ollero. Experimental results in multi-uav coordination for disaster management and civil security applications. *Journal of intelligent & robotic systems*, 61(1):563–585, 2011.

[147] W.M. McEneaney and BG Fitzpatrick. Control for uav operations under imperfect information. In *Proceedings First AIAA UAV Symposium, Portsmouth, VA*. Citeseer, 2002.

[148] T. M. McLain and R. W. Beard. Coordination variables, coordination functions, and cooperative timing missions. *AIAA Journal on Guidance, Control, and Dynamics*, 28(1):150–161, 2005.

[149] Fransisco S. Melo and Manuela Veloso. Decentralized mdps with sparse interactions. *Artificial Intelligence*, 175:1757–1789, 2011.

[150] N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[151] C. C. Moallemi and B. V. Roy. Consensus propagation. *IEEE Transactions on Information Theory*, 52(11):4753–4766, 2006.

[152] D. Monderer and L.S. Shapley. Potential games. *Games and economic behavior*, 14:124–143, 1996.

[153] Y. Mostofi. Decentralized communication-aware motion planning mobile networks: An information-gain approach. *Journal of Intelligent Robot Systems*, 56(2):718–740, 2009.

[154] J.M. Mulvey, R.J. Vanderbei, and S.A. Zenios. Robust optimization of large-scale systems. *Operations research*, 43(2):264–281, 1995.

[155] RA Murphey. Target-based weapon target assignment problems. *Nonlinear assignment problems: Algorithms and applications*, 7:39–53, 1999.

[156] R.A. Murphey. An approximate algorithm for a weapon target assignment stochastic program. *Nonconvex optimization and its applications*, 42:406–421, 2000.

[157] A. Nemirovski and A. Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2007.

[158] I. Nikolos, E. Zografos, and A. Brintaki. Uav path planning using evolutionary algorithms. *Innovations in Intelligent Machines-1*, pages 77–111, 2007.

[159] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[160] Reza Olfati-saber. Distributed Kalman filtering and sensor fusion in sensor networks. In *Network Embedded Sensing and Control*, volume 331, pages 157–167. Springer-Verlag, 2006.

[161] Reza Olfati-Saber, Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *IEEE Transactions on Automatic Control*, 95(1):215–233, January 2007.

[162] A. Olshevsky and J. N. Tsitsiklis. Convergence speed in distributed consensus and averaging. In *IEEE Conference on Decision and Control (CDC)*, pages 3387–3392, 2006.

[163] C.H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

[164] C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, pages 441–450, 1987.

[165] D. C. Parkes and L. H. Ungar. Iterative combinatorial auctions:theory and practice. In *Proceedings of the 17th National Conference on Artificial Intelligence*, 2000.

[166] K. Passino, M. Polycarpou, D. Jacques, M. Pachter, Y. Liu, Y. Yang, M. Flint, and M. Baum. Cooperative control for autonomous air vehicles. *Cooperative control and optimization*, pages 233–271, 2002.

[167] M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler. A stochastic and dynamic vehicle routing problem with time windows and customer impatience. *Mobile Networks and Applications*, 14(3):350–364, 2009.

[168] Sameera S. Ponda. Trajectory optimization for target localization using small unmanned aerial vehicles. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, September 2008.

[169] Sameera S. Ponda, Nisar Ahmed, Brandon Luders, Eric Sample, Tauhira Hoossainy, Danelle Shah, Mark Campbell, and Jonathan P. How. Decentralized information-rich planning and hybrid sensor fusion for uncertainty reduction in human-robot missions. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Portland, OR, August 2011 (AIAA Best Paper Award). (AIAA-2011-6238).

[170] Sameera S. Ponda, Luke B. Johnson, Han-Lim Choi, and Jonathan P. How. Ensuring network connectivity for decentralized planning in dynamic environments. In *Proceedings of the AIAA Infotech@Aerospace Conference*, St. Louis, MO, March 2011.

[171] Sameera S. Ponda, Luke B. Johnson, Alborz Geramifard, and Jonathan P. How. *Handbook of Unmanned Aerial Vehicles*, chapter Cooperative Mission Planning for Multi-UAV Teams. Springer, 2012 (to appear).

[172] Sameera S. Ponda, Luke B. Johnson, and Jonathan P. How. Distributed chance-constrained task allocation for autonomous multi-agent teams. In *American Control Conference (ACC)*, June 2012.

[173] Sameera S. Ponda, Luke B. Johnson, Andrew N. Kopeikin, Han-Lim Choi, and Jonathan P. How. Distributed planning strategies to ensure network connectivity for dynamic heterogeneous teams. *IEEE Journal on Selected Areas in Communications*, 30(5):861 –869, June 2012.

[174] Sameera S. Ponda, Joshua Redding, Han-Lim Choi, Jonathan P. How, Matt A. Vavrina, and John Vian. Decentralized planning for complex missions with dynamic communication constraints. In *American Control Conference (ACC)*, Baltimore, MD, July 2010.

[175] A. Pongpunwattana, R. Rysdyk, J. Vagners, and D. Rathbun. Market-based co-evolution planning for multiple autonomous vehicles. In *Proceedings of the AIAA Unmanned Unlimited Conference*, 2003.

[176] D.V. Pynadath and M. Tambe. The communicative multiagent team decision problem: analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16(1):389–423, 2002.

[177] Calyampudi Radakrishna Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, 37:81–89, 1945.

[178] C.E. Rasmussen. Gaussian processes in machine learning. *Advanced Lectures on Machine Learning*, pages 63–71, 2004.

[179] S. Rathinam, R. Sengupta, and S. Darbha. A resource allocation algorithm for multivehicle systems with nonholonomic constraints. *Automation Science and Engineering, IEEE Transactions on*, 4(1):98 –104, jan. 2007.

[180] J. Redding, A. Geramifard, A. Undurti, H. Choi, and J. How. An intelligent cooperative control architecture. In *American Control Conference (ACC)*, pages 57–62, Baltimore, MD, July 2010.

[181] J. D. Redding, N. Kemal Ure, J. P. How, M. Vavrina, and J. Vian. Scalable, MDP-based Planning for Multiple, Cooperating Agents. In *American Control Conference (ACC)*, June 2012.

[182] Joshua D. Redding. *Approximate Multi-Agent Planning in Dynamic and Uncertain Environments*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, February 2012.

[183] Wei Ren. Consensus based formation control strategies for multi-vehicle systems. In *American Control Conference (ACC)*, pages 6–12, June 2006.

[184] Wei Ren, R. W. Beard, and E. M. Atkins. Information consensus in multivehicle cooperative control. *IEEE Control Systems Magazine*, 27(2):71–82, April 2007.

[185] Wei Ren, R. W. Beard, and D. B. Kingston. Multi-agent Kalman consensus with relative uncertainty. In *American Control Conference (ACC)*, volume 3, pages 1865–1870, 8-10 June 2005.

[186] Wei Ren and R.W. Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, May 2005.

[187] A. Richards and J. How. A decentralized algorithm for robust constrained model predictive control. In *American Control Conference, 2004. Proceedings of the 2004*, volume 5, pages 4261–4266. IEEE, 2005.

[188] A. Richards and J. How. Decentralized model predictive control of cooperating UAVs. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 4, pages 4286–4291. IEEE, 2005.

[189] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House Publishers, Boston, MA, 2004.

[190] R.T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[191] R.O. Saber, W.B. Dunbar, and R.M. Murray. Cooperative control of multi-vehicle systems using cost graphs and optimization. In *American Control Conference, 2003. Proceedings of the 2003*, volume 3, pages 2217–2222. IEEE, 2003.

[192] A. Salman, I. Ahmad, and S. Al-Madani. Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems*, 26(8):363–371, 2002.

[193] D.J. Salmond. Mixture reduction algorithms for uncertain tracking. Technical Report 88004, Farnborough, UK: Royal Aerospace Est., 1988.

[194] S. Sariel and T. Balch. Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In *AIAA Workshop on Integrating Planning Into Scheduling*, 2005.

[195] K. Savla, E. Frazzoli, and F. Bullo. On the point-to-point and traveling salesperson problems for Dubins' vehicle. In *American Control Conference (ACC)*, pages 786–791, June 2005.

[196] Bruno Scherrer. Should one compute the Temporal Difference fix point or minimize the Bellman Residual? The unified oblique projection view. In *International Conference on Machine Learning (ICML)*, 2010.

[197] D. G. Schmale, B.R. Dingus, and C. Reinholtz. Development and application of an autonomous unmanned aerial vehicle for precise aerobiological sampling above agricultural fields. *Journal of Field Robotics*, 25(3):133–147, 2008.

[198] C. Schumacher, P. R. Chandler, and S. Rasmussen. Task Allocation for Wide Area Search Munitions via Network Flow Optimization. In *Proceedings of the American Control Conference*, pages 1917–1922, Anchorage AK, May 2002.

[199] S. Seuken and S. Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 17(2):190–250, 2008.

[200] S. Shebalov and D. Klabjan. Robust airline crew pairing: Move-up crews. *Transportation Science*, 40(3):300, 2006.

[201] T. Shima and S. J. Rasmussen. *UAV cooperative decision and control: challenges and practical approaches*, volume 18. Society for Industrial Mathematics, 2009.

[202] T. Shima, S. J. Rasmussen, and P. Chandler. UAV team decision and control using efficient collaborative estimation. In *American Control Conference (ACC)*, volume 6, pages 4107–4112, 8-10 June 2005.

[203] T. Shima, S.J. Rasmussen, A.G. Sparks, and K.M. Passino. Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms. *Computers & Operations Research*, 33(11):3252–3269, 2006.

[204] R Smith and R. Davis. Frameworks for cooperation in distributed problem solving. pages 61–70, January 1981.

[205] Daniel Southern. Human-Guided Management of Collaborating Unmanned Vehicles in Degraded Communication Environments. Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 2010.

[206] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *International Society for Adaptive Behavior*, 13(3):165–188, 2005.

[207] P. B. Sujit, D. Kingston, and R. Beard. Cooperative forest fire monitoring using multiple UAVs. In *IEEE Conference on Decision and Control*, pages 4875–4880, 2007.

[208] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. The MIT Press, 1996.

[209] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[210] Richard S. Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning (ICML)*, ICML '09, pages 993–1000, New York, NY, USA, 2009. ACM.

[211] A. Tahbaz-Salehi and A. Jadbabaie. On consensus over random networks. In *44th Annual Allerton Conference*, 2006.

[212] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[213] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[214] K. Tumer and D. Wolpert. A survey of collectives. *Collectives and the Design of Complex Systems*, pages 1–42, 2004.

[215] Aditya Undurti and Jonathan P. How. A Cross-Entropy Based Approach for UAV Task Allocation with Nonlinear Reward. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2010. AIAA-2010-7731.

[216] B. Upcroft, L.L. Ong, S. Kumar, M. Ridley, and T. Bailey. Rich probabilistic representations for bearing-only decentralised data fusion. In *FUSION 2005*, 2005.

[217] M. Valenti, B. Bethke, J. P. How, D. P. de Farias, and J. Vian. Embedding Health Management into Mission Tasking for UAV Teams. In *American Control Conference (ACC)*, pages 5777–5783, New York City, NY, 9-13 July 2007.

[218] E. Waltz and J. Llinas. *Multisensor data fusion*. Artech House Boston, London, 1990.

[219] Richard V. Welch and Gary O. Edmonds. Applying robotics to HAZMAT. In *The Fourth National Technology Transfer Conference and Exposition*, volume 2, pages 279–287, 2003.

[220] A. K. Whitten, H.-L. Choi, L. Johnson, and J. P. How. Decentralized task allocation with coupled constraints in complex missions. In *American Control Conference (ACC)*, pages 1642 – 1649, June 2011.

[221] Andrew K. Whitten. Decentralized planning for autonomous agents cooperating in complex missions. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, September 2010.

[222] C. W. Wu. Synchronization and convergence of linear dynamics in random directed networks. *IEEE Transactions on Automatic Control*, 51(7):1207–1210, 2006.

[223] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *International Symposium on Information Processing in Sensor Networks*, pages 63–70, April 2005.

[224] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas. Graph theoretic connectivity control of mobile robot networks. *IEEE Journal*, 99(9):1525–1540, 2011.

[225] H. Zhu, A. L. Swindlehurst, and K. Liu. Optimization of manet connectivity via smart deployment/movement of unmanned air vehicles. *IEEE Journal of Vehicular Technology*, 58(7):3533–3546, 2009.

[226] S. Zhu and M. Fukushima. Worst-case conditional value-at-risk with application to robust portfolio management. *Operations research*, 57(5):1155–1168, 2009.