

**CONTROL OF MULTI-AGENT NETWORKS: FROM
NETWORK DESIGN TO DECENTRALIZED
COORDINATION**

A Thesis
Presented to
The Academic Faculty

by

Philip Y. Twu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2012

CONTROL OF MULTI-AGENT NETWORKS: FROM NETWORK DESIGN TO DECENTRALIZED COORDINATION

Approved by:

Professor Magnus Egerstedt, Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Bo Hong
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Jeff Shamma
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Eric Feron
School of Aerospace Engineering
Georgia Institute of Technology

Professor Yorai Wardi
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Date Approved: 29 March 2012

*To my parents Jenn and Yu who never gave up,
my sunshine Cheryl, my dog Michael, and my friends.
Thank you for your motivation, patience, and understanding.*

PREFACE

This dissertation represents a culmination of my research in the GRITS (Georgia Robotics and Intelligent Systems) Lab at Georgia Tech under the supervision of Dr. Magnus Egerstedt from Fall 2008 to Spring 2012. In particular, it presents a suite of tools that I have developed which support the various stages of multi-agent design: ranging from initial network design, to local execution using decentralized coordination strategies. Together, the tools support a multi-agent system design methodology that is showcased through examples in three application domains: air traffic merging and spacing under the FAA's NextGen program, collaborative multi-UAV convoy protection in dynamic environments, and an educational tools for robotics. It is my firm belief that as autonomous and unmanned systems become more affordable, commonplace, and reliable in the near future, that the ideas which are presented here will contribute greatly in transitioning multi-agent systems from the lab setting to becoming an integral part of our everyday lives.

ACKNOWLEDGEMENTS

I want to thank my committee for providing helpful feedback on my research to make this dissertation possible. Moreover, I would like to thank all of my professors both as an undergraduate student at the University of Maryland, and as a graduate student at the Georgia Institute of Technology. These teachers have inspired and transformed me over the years from a naive student who knew so little about mathematics, science, and engineering, to the person I am today. Most importantly, I owe the most appreciation to my Ph.D. advisor, Dr. Magnus Egerstedt, who challenged me to think originally, and dared me to fall and pick myself back up again in the relentless pursuit of knowledge.

I would also like to thank the following organizations for partially supporting the work that is presented in this dissertation:

- Office of Naval Research through MURI Heterogeneous Unmanned Networked Teams (HUNT)
- Rockwell Collins Advanced Technology Center
- US National Science Foundation (NSF) through Grant # CCF 0820004.

TABLE OF CONTENTS

DEDICATION	iii
PREFACE	iv
ACKNOWLEDGEMENTS	v
LIST OF FIGURES	ix
SUMMARY	xii
I INTRODUCTION	1
1.1 Concepts	1
1.2 Background	5
1.2.1 Multi-Agent Networks	6
1.2.2 Network Design	8
1.2.3 Decentralized Controller Design	10
1.2.4 High-level Scripting of Control Strategies	11
1.2.5 Air Traffic Merging and Spacing	12
1.2.6 Collaborative Multi-UAV Convoy Protection	14
1.2.7 Educational Tools in Robotics	16
1.3 Goals	17
II NETWORK DESIGN PART 1: HETEROGENEITY	19
2.1 Complexity and Disparity	21
2.2 How can Heterogeneity be Quantified?	23
2.2.1 A New Measure of Heterogeneity	23
2.2.2 Conservation of Species within Heterogeneity-Maximizing Populations	27
2.2.3 Heterogeneity in Systems with Uniform Species	29
2.3 How Heterogeneous is a Multi-Agent System?	34
2.3.1 Examples: Heterogeneous Search-and-Rescue	37

III NETWORK DESIGN PART 2: EXPRESSIVENESS	40
3.1 System Dynamics	41
3.2 Controllability in Homogeneous Single-Leader Networks	43
3.2.1 Controllability of Single-Leader Networks	43
3.2.2 Optimally Reachable Target Points	47
3.2.3 Homogeneous Networks	49
3.3 When to Solve for Optimal Permutations	56
3.3.1 Optimal Permutations with $1D$ Agents	56
3.3.2 Optimal Permutations with nD Agents	58
3.4 Finding Suboptimal Permutations	59
3.4.1 Heuristic and Approximation Algorithms	60
3.4.2 Special Case: $1D$ Networks	60
IV GENERATING DECENTRALIZED CONTROLLERS	63
4.1 Optimizing Parameterized Modes	64
4.2 Application: Tracking Motions using Multi-Agent Networks	68
4.3 Simulation: Tracking a Drumline-Inspired Dance	70
V SCRIPTING DECENTRALIZED CONTROLLER SEQUENCES	74
5.1 Graph Process Specification Formulation	76
5.1.1 Networked System Representation	76
5.1.2 Atoms and Consistency	77
5.1.3 Modes and Composability	84
5.1.4 Graph Process Specifications	86
5.1.5 Executing Graph Process Specifications	88
5.2 Graph Process Specification Examples	90
5.2.1 Connectedness-Preserving Formation Control Laws	90
5.2.2 Line-Formation Decentralized Consistent Atom	92
5.2.3 Circle-Formation Decentralized Consistent Atom	95
5.2.4 Locally Executable GPS Example	97

5.3	Generating Decentralized Consistent Atoms Using Optimal Decentralization	102
VI	APPLICATIONS	106
6.1	Air Traffic Merging and Spacing	107
6.1.1	Problem Formulation	108
6.1.2	Feasibility Conditions	113
6.1.3	Real-Time Adaptation Algorithm	120
6.1.4	Simulations	125
6.2	Collaborative Multi-UAV Convoy Protection	134
6.2.1	Problem Formulation	135
6.2.2	Real-Time Adaptation Algorithm	137
6.2.3	Experimental Validation	144
6.3	Educational Tools for Robotics	149
6.3.1	Tasks	150
6.3.2	Educational Outcomes	157
VII	CONCLUSION	159
	REFERENCES	161
	VITA	173

LIST OF FIGURES

1	Example of a complex drumline-inspired multi-agent dance, where agents switch through a sequence of controllers that were each designed for a specific multi-agent motion. The locations of the agents are marked by O's and lines indicate their trajectory during the past 0.3 seconds.	3
2	Flowchart showing a general design flow for multi-agent systems that helps place the individual theoretical tools developed in this dissertation into a larger context. The stages are color-coded as follows: gray boxes correspond to processes which are not related to the design methodology, green are the main design stages, pink are the resources available, and blue are additional components used for real-time adaptation. The suite of tools developed in this dissertation fit into the stages outlined in red.	4
3	An example of a heterogeneous team of UAVs and AUVs searching a body of water for a target. Notice the inherent trade-off between having a homogeneous team of UAVs that can quickly cover only the surface, versus a heterogeneous team that can slowly cover the entire search area.	21
4	Different groups of colored marbles used to illustrate the importance of having both complexity and disparity in a heterogeneous system.	23
5	Plots showing the minimizing and maximizing distributions for entropy $E(p)$ and Rao's quadratic entropy $Q(p)$ in a multi-agent system with $M = 5$ uniform species, i.e., $d(i, j) = \alpha i - j $ for all $i, j \in \mathcal{M}$	26
6	Plot showing the heterogeneity maximizing distribution for H in a multi-agent system with uniform species, where $M = 5$. Notice that as stated in Theorem 2.2.2.1, all species have a nonzero portion of the total population assigned to it. Moreover, the shape of the distribution agrees with the description provided by Theorem 2.2.3.1.	35
7	Two examples of leader-invariant EEPs of a single-leader network, where V_1 is the vertex for the leader agent. (a) shows the trivial leader-invariant EEP, while (b) gives the maximal leader-invariant EEP. Since the two partitions are different, the network is not completely controllable.	44
8	The topology of the single-leader network in Example 3.2.3.1 is given in (a). (b) shows the closest the follower agents can reach $x_T = [1 \ 9 \ 10]^T$, while (c) shows the closest the follower agents can reach $Px_T = [9 \ 10 \ 1]^T$. Notice that the Px_T results in an error less than x_T and so Px_T is the better specification of the target configuration.	51

9	Convergence of cost J (tracking error) after performing steepest descent with Armijo step size on parameters and switching times for tracking a multi-agent drumline-inspired dance.	71
10	Simulation of $N = 21$ agents executing the drumline-inspired dance from Figure 1 using a decentralized controller sequence generated by the optimal decentralization algorithm. The locations of the agents are marked by O's with lines connecting them to their desired location marked by X's.	73
11	An illustration showing how the execution of an executable GPS with three modes can be viewed as a hybrid system.	90
12	The target points τ_{line} and the network topology (\mathcal{N}, E_{line}) for the line formation, with $N = 6$ and $\delta = 1$	93
13	The target points τ_{circle} and the network topology $(\mathcal{N}, E_{circle})$ for the circle formation, with $N = 6$ and $\delta = 1$	95
14	Simulation of agents executing the sequence of decentralized controllers scripted by the locally executable GPS_2 , as given in (80), for $N = 6$ and $\delta = 1$. The location of the agents are marked by O's and the lines indicate edges in the induced graph.	100
15	Aircraft on different legs of flight must merge while avoiding conflicts.	108
16	Top view of a two-track merging fork at the terminal phase of flight. .	109
17	Overhead view of ground track speed and path deviations during Phase II.	110
18	Ground track speed and aircraft separation during Phase III.	110
19	Diagram of Phase II used for proof of Theorem 3.3	117
20	Binary tree structure for merging multiple tracks.	119
21	Zoomed in illustration of binary tree structure for merging multiple tracks showing the distance needed between adjacent legs to avoid conflicts amongst aircraft.	121
22	Arrival time agreement and pairwise cost minimization.	128
23	Part one of a simulated binary tree structure merging three legs of air traffic onto a single terminal leg.	131
24	Part two of a simulated binary tree structure merging three legs of air traffic onto a single terminal leg.	132

25	Plots of separation distances amongst consecutive aircraft arrivals for each two-track merging fork's merge point in the binary tree merging simulation.	134
26	The multi-UAV convoy protection scenario with pop-up threats. . . .	136
27	Selection Policy in action. In this example, the threat is not a pop-up	141
28	System-level architecture: Team of UAVs and UGVs are connected through a WiFi router and tracked using a Vicon motion capture system.	145
29	Plots showing the trajectories of UAVs and the convoy which it protects as both a non-persistent and persistent threat are encountered.	147
30	Photos showing the convoy protection and threat neutralization as carried out by the hardware platform.	147
31	Plots showing the difference in path taken by the ground convoy before and after being informed about the presence of a persistent threat. . .	148
32	Plot of fuel consumption for the two UAVs illustrating that fuel usage is balanced while performing the convoy protection mission.	148
33	The virtual environment used for the multi-robot search and rescue final project. Students must design decentralized controllers to navigate a team of robots through all 6 waypoints in order, where each waypoint challenges the student to apply a different concept learned throughout the class.	151
34	Screenshots showing the students' solutions for solving all 6 waypoints in the multi-robot search and rescue mission.	154

SUMMARY

This dissertation presents a suite of design tools for multi-agent systems that address three main areas: network design, decentralized controller generation, and the synthesis of decentralized control strategies by combining individual decentralized controllers. First, a new metric for quantifying heterogeneity in multi-agent systems is presented based on combining different notions of entropy, and is shown to overcome the drawbacks associated with existing diversity metrics in various scientific fields. Moreover, a new method of controlling multi-agent networks through the single-leader network paradigm is presented where by directly exploiting the homogeneity of agent capabilities, a network which is not completely controllable can be driven closer to a desired target configuration than by using traditional control techniques. An algorithm is presented for generating decentralized control laws that allow for agents to best satisfy a desired global objective, while taking into account network topological constraints and limitations on how agents can compute their control signals. Then, a scripting tool is developed to aid in specifying sequences of decentralized controllers to be executed consecutively, while helping ensure that the required network topological requirements needed for each controller to execute properly are maintained throughout mode switches. Finally, the underlying concepts behind the developed tools are showcased in three example applications: distributed merging and spacing for heterogeneous aircraft during terminal approaches, collaborative multi-UAV convoy protection in dynamic environments, and an educational tool used to teach a graduate-level networked controls course at the Georgia Institute of Technology.

CHAPTER I

INTRODUCTION

Over the past decade, multi-agent systems have been demonstrated as an effective solution to many complex engineering problems (see Section 1.2 for examples). This dissertation presents a suite of design tools for multi-agent systems that address the following three areas: network design, decentralized controller generation, and synthesizing decentralized control strategies from individual decentralized controllers. Together, these tools fit into various stages of a general framework for multi-agent system design. However, each individual tool focuses on addressing a specific aspect of the design phase, and on overcoming issues associated with current related state of the art techniques. The concepts and performance of the developed tools are then showcased in three example applications: merging and spacing of heterogeneous aircraft during terminal approaches under the FAA’s NextGen program, collaborative multi-UAV convoy protection in dynamic hostile environments, and an educational tool for teaching networked controls at the graduate level. This chapter will act as an introduction by describing the main concepts behind each of the theoretical tools that will be developed in this dissertation, familiarizing the reader with relevant background material, and stating the goals of the research.

1.1 Concepts

Designing a multi-agent system for performing a specific task requires that many design parameters be taken into consideration. For example, suppose it is desired to design a network of agents that can perform the drumline-inspired dance shown in Figure 1. In such a scenario, the design parameters may include the choice of agents that the system will comprise of, the underlying network topology, the decentralized

control laws used by the agents, and the switching strategy used for the controllers.

Attempting to take all these design parameters into consideration at once may prove to be overly cumbersome in many situations. Instead, to make the task more tractable, the design procedure can be partitioned into a series of specialized stages. Using such a modular approach not only simplifies the design task at each stage, but also promotes the reusability of design tools. An example of one such design methodology for multi-agent systems, which will act as a road map to help place the theoretical tools that will be developed in this dissertation in to a larger context, is shown in Figure 2. In the flowchart, gray boxes correspond to processes which are not related to the design methodology, green are the main design stages, pink are the resources available, and blue are additional components used for real-time adaptation. The suite of tools that will be developed in this dissertation will fit into the stages outlined in red.

The stages shown in the flowchart can be summarized as follows. First, when given a multi-agent task specification, it may be necessary to decompose it into a series of subtasks in order to reduce the complexity of controller design. Simultaneously, one has to design an appropriate multi-agent system, from the resources which are available, that is capable of carrying out all of the subtasks. Here, the relationship between the system design parameters (e.g., agent composition and network topology) and the effect on the system's overall capabilities (e.g., controllability properties) must be considered. To do so, an understanding of the connection between a multi-agent's heterogeneity and its expressiveness is required. If it is not possible to design a system using the resources available that can accomplish each of the subtasks, then the subtasks themselves must be re-evaluated. Therefore, the two design steps form an iterative process that outputs the following: a sequence of subtasks, and a multi-agent system specification.

Next, it is necessary to specify a decentralized control strategy allows for the

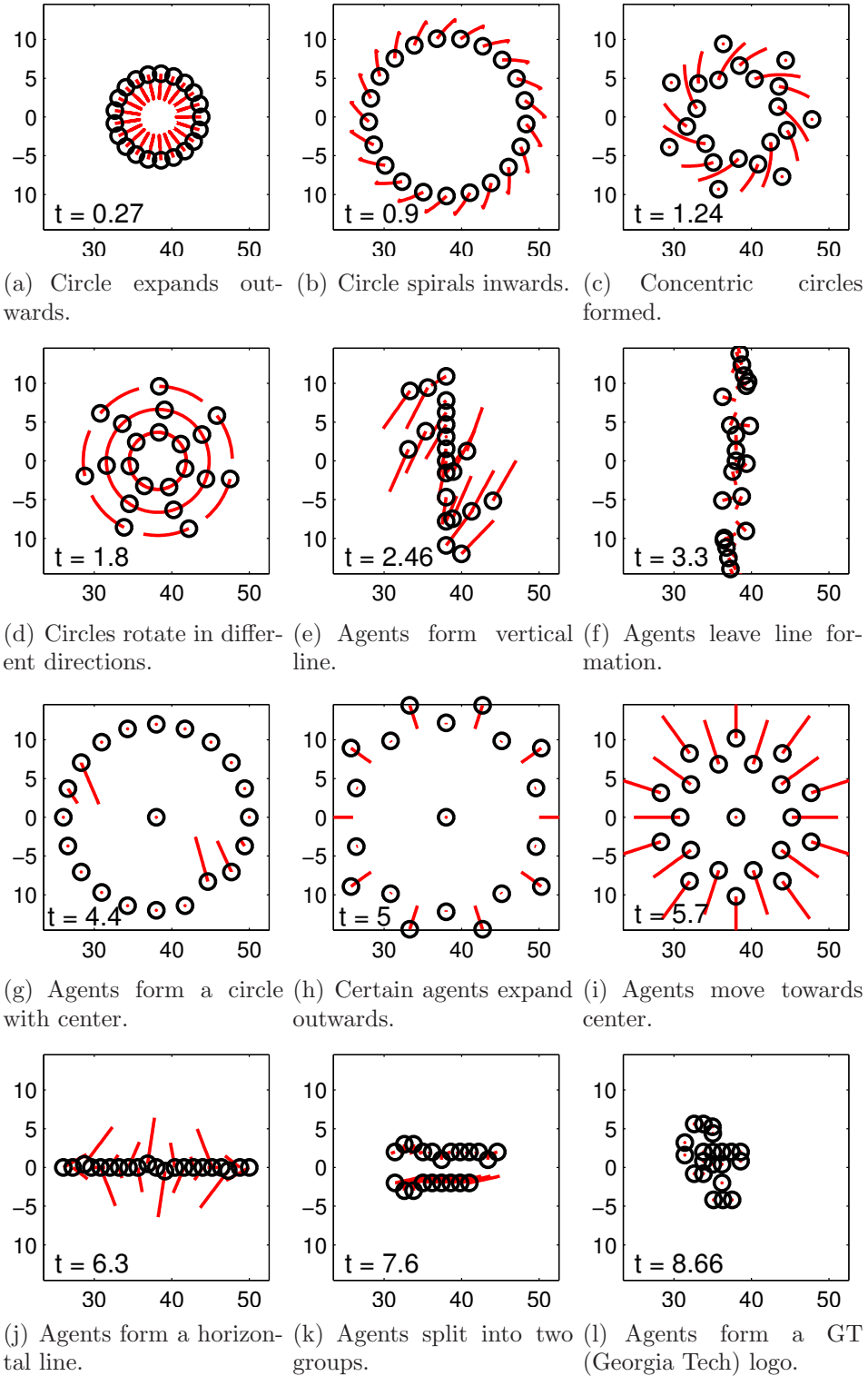


Figure 1: Example of a complex drumline-inspired multi-agent dance, where agents switch through a sequence of controllers that were each designed for a specific multi-agent motion. The locations of the agents are marked by O's and lines indicate their trajectory during the past 0.3 seconds.

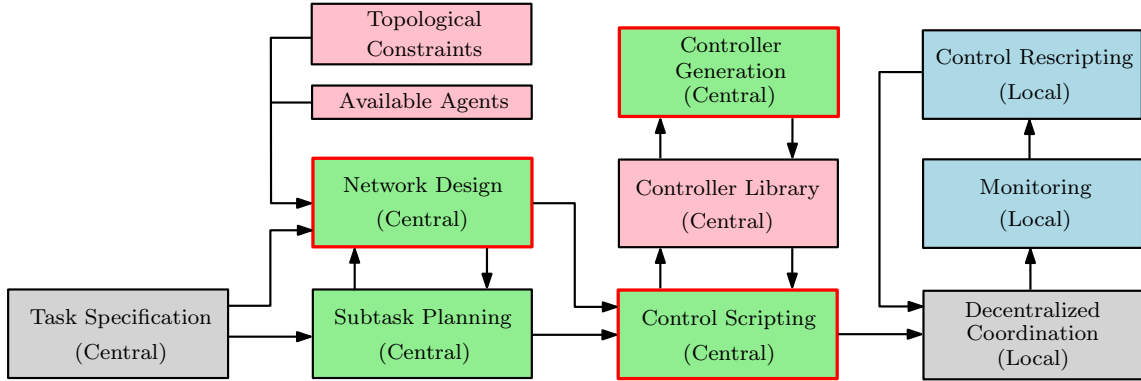


Figure 2: Flowchart showing a general design flow for multi-agent systems that helps place the individual theoretical tools developed in this dissertation into a larger context. The stages are color-coded as follows: gray boxes correspond to processes which are not related to the design methodology, green are the main design stages, pink are the resources available, and blue are additional components used for real-time adaptation. The suite of tools developed in this dissertation fit into the stages outlined in red.

specified multi-agent system to carry out each of the subtasks. In particular, high-level control scripting tools that specify a sequence of decentralized control laws and an associated switching strategy can be used. Specific controllers for agents to accomplish each subtask with can be queried from an available library of controllers. Alternatively, if a particular controller cannot be found in the library, one can be generated using decentralized controller generation algorithms and then inserted into the library. Note, however, that agents may be switching abruptly between executing different decentralized controllers, each with their own set of required network topological operating conditions (e.g., network connectivity). Therefore, the scripting tool must enforce appropriate guard conditions and constraints on the mode ordering such that each controller’s required operating conditions are satisfied. If these requirements cannot be met, then the script must be modified accordingly and checked again in an iterative process that outputs the following: a sequence of decentralized controllers, and a set of locally-checkable guard conditions for the agents to use when transitioning between modes.

Finally, having designed a multi-agent system and a decentralized coordination

strategy, agents can start executing the originally specified task. However, during execution, agents should always locally monitor the environment in order to detect any changes in the operating conditions. In the event that real-time adaptation is necessary, agents can perform a local rescripting of their control strategy by changing the decentralized control law it is using to perform a particular subtask with. However, additional care must be made to ensure that such a change does not affect the performance of other agents.

The previous paragraphs have outlined the main concepts a general design methodology for multi-agent systems in Figure 2, that will act as a road map to place the research presented in this dissertation into a single unifying picture. In particular, this dissertation will present theoretical tools that fit into the three stages outlined in red: network design, decentralized controller generation, and scripting controller sequences. Moreover, the application of the methodology as a whole to the engineering of multi-agent systems will be showcased in three examples. Before clearly stating the technical goals of this dissertation, however, a literature review will first be presented to familiarize the reader with any relevant background information.

1.2 Background

This section will present background information on material that is presented in the various chapters of this dissertation. Starting with an overview of research in multi-agent networks, supporting literature will then be provided for the three types of tools that will be developed: network design, decentralized controller generation, and high-level scripting of controller sequences. Then, additional background information will be provided for the three example applications of the design methodology: air traffic merging and spacing during terminal approaches, collaborative multi-UAV convoy protection in dynamic and hostile environments, and educational tools for robotics.

1.2.1 Multi-Agent Networks

The research presented in this dissertation is centered around the decentralized control of multi-agent networks. The main idea behind decentralized control is very similar to that of distributed algorithms (e.g., [63]) in computer science. In both fields, individual entities within a system must make use of locally available information and resources to accomplish some global goal. However, decentralized control of multi-agent networks differs in that the goal is to control a dynamical system to a set of desired states, as opposed to performing some computation. Some of the earliest and most influential work in multi-agent systems came from an attempt to understand and recreate naturally occurring phenomena. For example, the Boids model presented in [100] was an attempt to reproduce the bird flocking behavior observed in nature for the movie animation industry. Realistic-looking simulated flocks were created by having simulated birds follow a set of simple navigation rules for separation, alignment, and cohesion, based on each bird’s local perception of the environment. Likewise, the Vicsek model from [129] reproduced the coherent alignment behaviors observed amongst particles by using update rules which set the heading of each simulated particle equal to the average heading of itself and those within a certain proximity.

In both the Boids and Vicsek models, the observed flocking and alignment behaviors resulted from agents in the system reaching a “consensus” on their heading angles and speeds. In general, having agents reach consensus on some quantity of interest lies at the heart of many multi-agent coordination strategies. It is of no surprise then, that a vast amount of research has been conducted on consensus protocols for multi-agent networks (see [83] for a comprehensive guide). Pioneering work in the development of consensus control laws include [44, 84, 99]. In particular, [44] uses tools from algebraic graph theory (i.e., [40]) to analyze the particle heading alignment algorithm from the Vicsek model, for systems with undirected network topologies. [84]

presents convergence analysis results on consensus protocols for systems with directed and dynamic network topologies, as well as for undirected networks in the presence of time delays. Further important convergence results are derived in [99] for consensus protocols in systems with directed and dynamic network topologies.

Besides reaching consensus, multi-agent research is also interested in having agents accomplish other tasks in a decentralized manner. Examples include having agents maintain formations, mimic flocking and swarming behavior, and spread out to cover an area of interest. The goal of formation control in multi-agent systems is to design decentralized controllers that safely and efficiently maneuver agents into a desired configuration. Some influential work in the area of formation control include [19, 20, 33, 35, 38, 51, 57, 59, 117]. Flocking and swarming behaviors have long been of interest to biologists who seek to understand how groups of animals, fish, and insects can perform collective decision making without a clearly designated leader. Engineers are also interested in understanding this phenomena to use it for practical applications, such as performing foraging or search-and-rescue missions with a team of mobile robots. Important recent literature on decentralized flocking and swarming include [27, 28, 30, 58, 61, 82, 116]. Coverage algorithms are concerned with having agents spread out and occupy an area of interest using only locally available information. For example, in the case of mobile sensor networks, one would like for agents to span an area such that the probability of any agent detecting a randomly occurring event is maximized. Notable recent work on coverage control includes the research presented in [26, 31, 67, 71, 78, 109]. The work highlighted thus far is but a subset of the rich and diverse research published on the decentralized control of multi-agent networks. For a more comprehensive guide, the reader is referred to [72].

1.2.2 Network Design

To design a multi-agent system for performing a particular task, one must start out with choosing both the agent composition and the network topology of the system. Agent heterogeneity is something that is used often in practical applications of multi-agent systems (e.g., [10, 41, 52, 91, 111]), but is in general not a well understood property. Because of this, there have been many attempts to quantify heterogeneity in various scientific fields. Within economics, the Atkinson, Gini, and Theil indices described in [29] are used to measure income inequality across a population. Within biology, metrics such as the Berger-Parker, Shannon, and Simpson indices from [112] quantify the diversity of a population solely based on the percentage of the total population belonging to each species. Rao’s quadratic entropy was proposed in [98] as a means to measure species diversity, while also taking species differences into account. However, [89] showed that in certain scenarios, maximizing Rao’s quadratic entropy requires that certain species be eliminated from the system completely, which disagrees with common intuition on diversity. Finally, in the multi-agent robotics community, hierarchical social entropy was presented in [11] as a way to quantify the diversity of a multi-robot system. Hierarchic social entropy considers a species of agents as a cluster resulting from hierarchical clustering. Diversity is then measured by examining how the system’s entropy varies as the clustering threshold increases.

The research discussed in this dissertation on heterogeneity will address the unanswered questions of “what is heterogeneity?”, “how can heterogeneity be quantified?”, and “how heterogeneous is a multi-agent system?”. To do so, a new metric is developed for measuring heterogeneity in multi-agent systems based on the concept of complexity and disparity in agent distributions across species. The metric is shown to avoid many of the problems encountered with the discussed existing metrics as an accurate measure for diversity. Moreover, by using the notion of a task-space, agents with different capabilities can be compared on common ground.

Having defined what it means for a multi-agent system to be heterogeneous, the next step is to explore the influence of heterogeneity (or lack thereof) on the expressiveness of a system. In particular, the controllability properties of leader-follower networks will be considered where a user may influence agents in the system through controlling a leader agent. In [115], necessary and sufficient conditions for the controllability of single-leader networks were presented in the form of algebraic tests. A graph-theoretic interpretation of the necessary conditions for single-leader controllability was given in [94] using equitable partitions. The controllability analysis was also extended to the case when multiple leaders exist in the network. Building off those results, [68] used the concept of relaxed equitable partitions to present a graph-theoretic interpretation of both the necessary and sufficient conditions for controllability of a single-leader network. Finally, the notion of structural controllability was investigated in [60], which considers the controllability of a multi-agent network when edge weights may be chosen freely.

Existing results on the controllability of single-leader networks fit nicely into the context of heterogeneous agents since it is implied in the analysis that each agent has a unique role in the network. Hence, the results describe when given a set of target points, whether it is possible to control the network such that each agent goes to their corresponding target. However, in this dissertation, analytical results will be presented from [121, 124] on the controllability analysis of single-leader networks where the homogeneity of agents is taken into account explicitly. When controlling homogeneous agents to a set of target points, if all that matters is the presence of an agent at each target point, then the labels on the targets may be permuted freely while still specifying the same configuration. However, some permuted target points may be closer to the multi-agent system's reachable subspace than others. Thus, controllability for homogeneous multi-agent systems changes from a traditionally-viewed point-to-point property, which is more fit for analyzing heterogeneous systems,

to now a point-to-set property of the system.

1.2.3 Decentralized Controller Design

Having explored the relationship between the heterogeneity and expressiveness of a network, the next step is to delve into how to design decentralized controllers for the agents to coordinate and accomplish some task in mind. In general, two distinctly different approaches exist for designing decentralized controllers. The first approach can be thought of as a bottom-up approach. In the bottom-up approach, local controllers for individual agents are designed first. Analysis is then performed to show the global properties which are exhibited when agents in the network execute the controllers. Many of the decentralized controllers mentioned earlier for consensus (e.g., [44, 48, 84, 99, 116]), formation control (e.g., [35, 117]), and swarming (e.g., [27, 58, 61, 82]) were designed using this paradigm. Conversely, the top-down approach involves first specifying a global performance metric, and then investigating when the resulting optimal controller is in fact decentralized. Examples of work that follows this view include [13, 76, 96, 102, 132].

This dissertation presents a method to generate sequences of decentralized controllers to track desired multi-agent motions, based on the optimal decentralization algorithm from [123, 125]. The algorithm bridges existing top-down and bottom-up approaches to designing decentralized controllers by first specifying a multi-agent tracking task in the form of a global performance metric. In addition to the metric, parameterized constraints are specified that describe what constitutes a decentralized controller for the system. An optimization problem is then solved to find the decentralized controllers' parameters, as well as the mode switch times [34], so as to make agents best track the desired motion.

1.2.4 High-level Scripting of Control Strategies

Having addressed how to design decentralized controllers for agents, the next step is to combine them together into more complex coordination strategies. To do so, specialized controllers for basic subtasks are oftentimes combined with switching logic. Thus, the multi-agent system becomes a hybrid system, where low-level continuous dynamics are coupled with high-level discrete mode-switches. A sample of the existing literature on the control of hybrid systems includes [4, 7, 15, 17, 43, 55, 75, 108].

There are many examples of multi-agent control using high-level abstraction-based techniques. For instance, embedded graph grammars (EGG) have been shown in [70, 110] as a useful tool for specifying rules on how agents should choose from a set of local controllers. [54] presents an architecture for abstracting the essential features of a multi-agent system and using linear temporal logics (LTL) to specify group-level goals. Model-checking programs are then used to generate trajectories that accomplish the goals, which are then mapped to provably-correct low-level control laws for individual agents to execute. Motion description languages (MDL), as presented in [18, 64, 66], specify motion programs to be executed by a system through sequences of controller and interrupt conditions. In particular, the MDL_n framework in [66] specifies a language that allows motion programs to be written with embedded network information requirements.

The Graph Process Specification (GPS) framework from [125, 126] is presented in this dissertation as a scripting tool for specifying multi-agent motion programs, similar to how a MDL works, by stating a sequence of decentralized controllers and locally-checkable guard conditions. Moreover, the scripting tool simultaneously ensures that the network topological operating conditions required by each controller (e.g., network connectivity) are always satisfied. To do so, atoms are used to explicitly state the operating conditions needed for agents to begin using a decentralized controller, and the conditions guaranteed upon its termination through the use of a locally checkable

guard condition. Complex controller sequences can then be constructed by stringing together atoms in a manner closely related to the mode sequencing problem for hybrid systems (e.g., [2, 7]), except with constraints limiting which atom pairs can be executed consecutively. Moreover, GPS also allows for additional interrupt conditions to be specified that determine when agents switch from executing one mode in the sequence to another. Such a scenario is reminiscent of the mode scheduling problem (e.g., [9]), where optimization occurs over both the mode sequence and switch times between the modes (e.g., [106, 107, 133, 134]).

1.2.5 Air Traffic Merging and Spacing

The design methodology shown in Figure 2 will be showcased in a series of three example applications. The first application is a solution for merging and spacing heterogeneous aircraft during terminal approaches in support of the FAA’s Next Generation Air Transportation System (NextGen) program. NextGen is the FAA’s vision to address the impact of air traffic growth by increasing the National Airspace System’s capacity and efficiency, while improving the safety and reducing environmental impacts [80]. It is expected that under NextGen, the so-called performance-based navigation (PBN) will allow aircraft to fly negotiated trajectories, thereby changing the air traffic controller’s tasks from clearance-based control to trajectory management. One of NextGen’s goals is to explore improvements in terminal area operations, namely the automatic merging and spacing of incoming flight paths, in order to increase the air traffic capacity of the terminal phase and save fuel by reducing extraneous flight maneuvers, e.g., holding patterns. Current systems completely rely on air traffic controllers to safely route aircraft. As a result, conflicts in merging routes are often identified too late and aircraft are asked to hold or redirect in order to wait for an opening, thus creating an excessive separation between the aircraft.

Safe and efficient merging of air traffic in support of the FAA’s NextGen program is an active area of research and is the subject of a few large-scale tests of systems developed based on Automatic Dependent Surveillance-Broadcast (ADS-B) information. ADS-B, a crucial component of NextGen, relays highly accurate traffic information between equipped aircraft and a network of satellites and ground stations [104]. SafeRoute, which is implemented on UPS aircraft, is an example of a centralized and large-scale ADS-B based technology. Air traffic controllers instruct the pilot to follow a particular aircraft, while an on-board system actively computes and displays a recommended aircraft speed such that a safe distance is maintained with the leading aircraft and safe merging is guaranteed at the merge points [8]. In Point Merge, another centralized merging and spacing solution, aircraft approaching the terminal area achieve the desired separation by flying on one of the vertically spaced sequencing legs to extend the flight path as necessary [37]. The National Aeronautics and Space Administration (NASA) is also actively involved in air traffic management research [36]. NASA’s Aviation Systems Division is focusing on hi-flow airports [128], high density en route operations, and automated separation assurance by using trajectory based tactical air traffic management [69].

A central theme in air traffic management research lies in the problem of conflict resolution amongst aircraft. Consequentially, many research efforts have been made in addressing this subject. [118] presents a game-theoretic approach for conflict resolution of noncooperative aircraft. [65] provides sufficient conditions for stable conflict avoidance of two intersecting aircraft flows. [95] gives a decentralized deconfliction algorithm based on artificial potential functions. [131] uses the bargaining technique of Monotonic Concession Protocol to detect and pseudo-optimally resolve conflicts. [103] suggests a slot-based model where en-route traffic select an available slot and then maintain its positioning in the traffic flow, hence guaranteeing safety-of-flight.

The merging and spacing solution presented in this dissertation is based off of the

work in [23, 24, 120] on distributed merging and spacing of heterogeneous aircraft (i.e., where spacing distances depend on the type of aircraft present) during the terminal phase of flight in support of NextGen. A set of feasibility conditions are identified on the geometry and operating conditions of merging forks such that safe merging is guaranteed. Under such conditions, merging aircraft can then negotiate using the ADS-B protocol through dual decomposition (e.g., [86, 97]) to agree on merging times and flight plans that minimize a pairwise cost, while maintaining proper inter-aircraft separation based on the aircraft type. Optimal flight trajectories are then generated locally with parameterized decentralized control laws using concepts from the optimal decentralization tool presented in this dissertation.

1.2.6 Collaborative Multi-UAV Convoy Protection

The second example application of the design methodology from Figure 2 will be for a collaborative multi-UAV convoy protection scenario in a dynamic and potentially hostile environment. Due to the recent upsurge in the availability of autonomous vehicle technology in military operations, many efforts have been made to incorporate teams of unmanned autonomous vehicles in human-led intelligence, surveillance, and reconnaissance (ISR) missions. Lockheed Martin’s Survivability Planner Associate Rerouter (SPAR) program (i.e., [45, 46, 114]) lets the co-pilot of an Apache Longbow helicopter command a team of UAVs to perform sensing-based tasks with the aid of multi-sensor data fusion algorithms. Northrop Grumman’s Heterogeneous Airborne Reconnaissance Team (HART) program (i.e., [1, 81, 85]) lets ground troops put in sensing-based requests for an accompanying team of UAVs, that then returns geo-registered video imagery mosaics with variable levels of detail based on the available bandwidth. Draper Labs’ Risk-Aware Mixed-Initiative Dynamic Replanning (RMDR) system (i.e., [113, 130]) lets human operators oversee a heterogeneous team of UUVs and USVs during a surveillance mission by making decisions on sensed

targets and specifying timing constraints for tasks.

A common problem faced by all of the presented approaches to integrating unmanned vehicle teams into human-led ISR tasks is the need to assign the vehicles to targets in an efficient manner. For example, given a single UAV and a set targets to be visited, together with a cost associated with traversing between the targets, the problem of selecting the order in which the targets should be visited (and cleared) is a variant of the well-studied NP-hard traveling sales person (TSP) problem [21, 50]. If multiple UAVs are present (greater than or equal to the number of targets), then the scenario reduces to the matching problem [88], where one simply has to decide which target is visited by which UAV. The matching problem can be solved in cubic time (in the number of targets) by using the Hungarian algorithm [56], as is demonstrated in [47, 74].

In most realistic ISR scenarios, however, there are most likely fewer UAVs than targets. Therefore, the NP-hardness still applies (e.g., [5, 90]) and must be dealt with directly. Numerous algorithms for approximating and addressing the TSP problem in a computationally tractable manner have been proposed. These algorithms generally fall under two categories: greedy algorithms, and auction-based game-theoretic algorithms. Greedy approaches (i.e., [14, 93, 101, 105]), oftentimes have agents focus on locally minimizing an instantaneous cost as opposed to over the entire time horizon, thus yielding only a suboptimal solution. Auction and game-theoretic based approaches (i.e., [6, 25]) let agents bid on which tasks they want to be assigned to and typically require a significant amount of information passing. [92] presents a solution that falls between the two categories by focusing on the feasibility of the solution over limited time horizons, rather than the minimization of any particular cost function, to obtain decentralized suboptimal solutions.

The collaborative multi-UAV convoy protection algorithm presented in this dissertation is based off of the work in [119] and avoids having to solve the TSP problem

all together. Instead, the order in which threats must be visited is assumed to be known a priori, sorted by the time it takes to be encountered by the convoy when moving along its planned path. Each UAV is capable of clearing the threat but only a single UAV is required for the clearing task, while the others should remain with the convoy. The assignment task therefore follows the same concept as the homogeneous multi-agent controllability design tools discussed in this dissertation. Here, the multi-UAV team must be controlled to a permutation of the target configuration, where one target point is located above the threat to be cleared, and the other target points are above the convoy. However, instead of minimizing the distance between the UAVs and the target configuration, the assignment is made so as to balance the UAVs' fuel consumption so as to maximize the duration of time in which the UAVs are operational. Finally, the convoy is also allowed to change its path if a threat cannot be cleared by the UAVs.

1.2.7 Educational Tools in Robotics

The third and last example application of the design methodology in Figure 2 is an educational tool for robotics education. In general, theoretically-oriented courses in control theory tend to focus too heavily on mathematical theory and proofs, whereas dedicated laboratory courses tend to be too time consuming and resource-intensive to maintain. To compromise, some courses have begun adopting smaller-scaled “take-home” labs (e.g., [12, 32, 39, 42]), where traditional lectures in controls are supplemented by portable and inexpensive robotics-based experiments that let students explore and implement the concepts learned. However, extending take-home labs to courses in multi-robot coordination has proven quite difficult since both the cost of equipment and probability of hardware failure increases with the number of robots involved [53]. Instead, simulated “virtual environments” (e.g., [22, 73, 79, 127]) offer the better alternative due to their low cost, reconfigurability, and ease of transport.

This dissertation presents a virtual environment for multi-robot experimentation based off of the work presented in [122], that has been integrated into the graduate-level networked controls course ECE8823 at the Georgia Institute of Technology every year since the Fall 2010 semester. Using the design methodology from Figure 2 as a road map, students must design decentralized control strategies for a team of simulated robots to perform a search-and-rescue mission in a simulated asteroid terrain. The search-and-rescue mission is broken into a series of 6 subtasks: rendezvous, squeezing through a tunnel, navigating through obstacles, performing sensor coverage, splitting and merging, and getting into a formation. To complete the mission, students must design decentralized control laws for agents to accomplish each subtask with. Moreover, controller scripting ideas from the Graph Process Specification (GPS) tool presented in this dissertation are used to add in guard conditions and perform mode sequencing operations, so as to ensure that the entire decentralized controller sequence can be executed consecutively by agents to complete the mission.

1.3 Goals

The goal of this dissertation is to develop a suite of theoretical tools that aid in the design of multi-agent systems and overcome issues associated with related state of the art techniques. In particular, Chapter 2 addresses the lack of a unifying understanding and definition for heterogeneity in the context of multi-agent systems. The research presented in this chapter aims to define a unifying metric for quantifying diversity which overcomes problematic issues that existing diversity metrics suffer from, and is directly applicable to systems where individual agents may be capable of accomplishing different subsets of tasks from a finite discrete set of tasks. Chapter 3 deals with the problem of quantifying how closely the states of agents in a network can be controlled to a set of desired target points when it is not possible to directly interact with each of the agents. In particular, the focus is to exploit homogeneity

in the agents' capabilities so as to improve upon the current limitations of controllability when a single-leader control paradigm is used and the resulting system lacks complete controllability. Chapter 4 looks at the problem of designing decentralized control laws that can be uploaded to agents in a network which will allow for them to best accomplish some desired global objective. The goal is to develop an algorithm so as to generate decentralized control laws so as to control agents to minimize a global cost, while explicitly taking into consideration limitations on the inter-agent information flow and computational capabilities of the agents. Chapter 5 considers the issue where if the network topology is dynamic, arbitrary decentralized control laws might not be able to be executed back to back successfully due to the possibility of having the network topological requirement associated with a controller being violated upon switching modes. The objective is to create a scripting tool that can help control strategy designers deal with satisfying network topological requirements when specifying a sequence of decentralized controllers that is to be executed back to back, while still maintaining focus on the high-level control design task of mode sequencing and scheduling. Finally, the goal of Chapter 6 is to, through a series of example applications, showcase the performance and modifications that are needed so as to use the ideas behind the theoretical tools that were presented in this dissertation in a practical setting.

CHAPTER II

NETWORK DESIGN PART 1: HETEROGENEITY

The first set of tools to be developed in this dissertation will be under the theme of network design. In general, these tools will provide the designer of a multi-agent system with intuition and insight as to how the heterogeneity of agents (or lack thereof) in the network influences what the network is capable of doing, i.e., its expressiveness. Having access to such information is useful to engineering design for two reasons. The first is that when given a set of tasks, one would like to design a multi-agent system which is capable of accomplishing them. An example of how one would use network design tools in this manner is illustrated in the multi-agent design methodology flowchart in Figure 2. Alternatively, in the event where the task is not known a priori, one should design the system to be as expressive as possible in order to be prepared for a multitude of scenarios. Such a design strategy is useful, for example, when designing a team of search-and-rescue robots that will be operating in a highly dynamic environment. This dissertation will explore the subject of network design from two different vantage points: heterogeneity in Chapter 2, and its implications on a system's expressiveness in Chapter 3.

Heterogeneous multi-agent systems have been used in a wide-variety of applications (see Section 1.2.2 for examples). Surprisingly, the effects of incorporating heterogeneity as a design variable for a system has been mostly unexplored in the existing literature. To illustrate the usage of heterogeneity as a design parameter, consider the need to create a team of unmanned vehicles that can quickly and effectively search a certain enclosed space for a threat. A homogeneous team of UAVs maybe able to search the surface of a lake for a threat, but is unable to venture

into any underwater caverns. A heterogeneous team of half UAVs and half AUVs, as is illustrated in Figure 3, may take longer to search the surface of the lake due to there being fewer UAVs. However, the presence of AUVs means that it is capable of searching the lake more thoroughly than the homogeneous team. One can see from this example that varying heterogeneity as a design parameter can potentially have profound implications on a system’s capabilities.

One of the reasons why there lacks a deeper understanding of the effects of heterogeneity on a multi-agent system stems from the fact that existing metrics for heterogeneity suffer from various problems. As a result, there is no universally agreed-upon definition of heterogeneity across scientific disciplines. Examples of the wide variety of existing diversity metrics that are available include the Atkinson, Gini, and Theil indices used to measure income inequality in economics, the Berger-Parker, Shannon, and Simpson indices used to measure biodiversity in ecology (see literature review in Section 1.2.2 for more examples), and hierarchic social entropy in robotics. Therefore, although it is simple to label a multi-agent system as being heterogeneous, fundamental questions such as:

1. What is heterogeneity?
2. How can heterogeneity be quantified?
3. How heterogeneous is a multi-agent system?

remain unanswered. In this chapter, we take a step back from current research efforts on heterogeneous systems and instead start by answering the three fundamental questions which were posed. *Only after a universal definition of heterogeneity has been established, can its effects on a multi-agent system be uncovered systematically.* The goal is to develop a unifying metric for defining and quantifying heterogeneity in multi-agent systems that can overcome the issues which existing diversity metrics

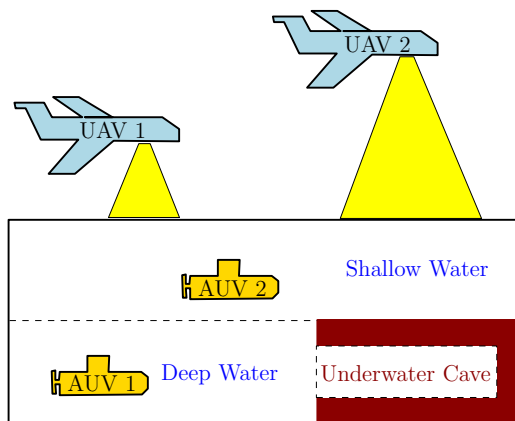


Figure 3: An example of a heterogeneous team of UAVs and AUVs searching a body of water for a target. Notice the inherent trade-off between having a homogeneous team of UAVs that can quickly cover only the surface, versus a heterogeneous team that can slowly cover the entire search area.

suffer from. Moreover, the metric should be practical for multi-agent applications requiring agents to work together and accomplish a finite set of tasks by discriminating agents based on their abilities to carry out each those tasks.

This chapter is organized as follows: by using existing work on diversity metrics from various scientific fields in Section 1.2.2 as a starting point, Section 2.1 sets out to understand what heterogeneity is by exploring the concepts of complexity and disparity. From this, Section 2.2 will propose a new metric for quantifying heterogeneity in multi-agent systems that overcomes many of the issues which existing diversity metrics suffer from. Finally, in Section 2.3, this metric will be specialized for the case of describing multi-agent systems where the agents are able to accomplish different tasks.

2.1 Complexity and Disparity

We will illustrate what characteristics of a system are required for heterogeneity through a simple example.

Example 2.1.0.1. Consider the task of picking handfuls of colored marbles out of a bag. For simplicity, let the differences amongst the types (or species) of marbles be

given by the difference in their shade of darkness. Suppose the first batch of marbles picked (Group A) was all white, as shown in Figure 4(a). Despite lacking a concise definition of heterogeneity at this point, it is clear that Group A is not heterogeneous at all. Now, suppose two more batches (Groups B and C) of marbles are picked from the bag as shown in Figures 4(b) and 4(c) respectively. Both are clearly more heterogeneous than Group A, but in different ways.

Group B is more heterogeneous than Group A since its marbles are distributed evenly amongst three different types, whereas all the marbles in Group A are of the same type. We will describe how distributed the marbles are to different types as the group's "complexity". Looking now at Group C, we see that its marbles are only distributed evenly amongst two types and so it is more complex than Group A but less so compared to Group B. However, notice that while Group B has more types present, the colors of the marbles are only slight variations of one another. Compare this to what is seen in Group C where different types of marbles have very distinct colors. We will describe this notion of how distinct the marbles are from one another as the group's "disparity".

Using these terms, we see that the homogeneous Group A has neither complexity nor disparity. Group B is more heterogeneous than Group A because it exhibits higher complexity. On the other hand, Group C is more heterogeneous than Group A because it shows higher disparity. Complexity and disparity thus serve as two key characteristics that must be present in a heterogeneous system. However, oftentimes an increase in one may decrease the other. Therefore, to maximize heterogeneity, a system must balance the two properties as shown in Group D of Figure 4(d).

The previous example showed that both complexity and disparity must be present in a heterogeneous system. Moreover, a heterogeneous system must strike a balance between the two oftentimes competing properties. Based on these observations, we propose the following relationship:

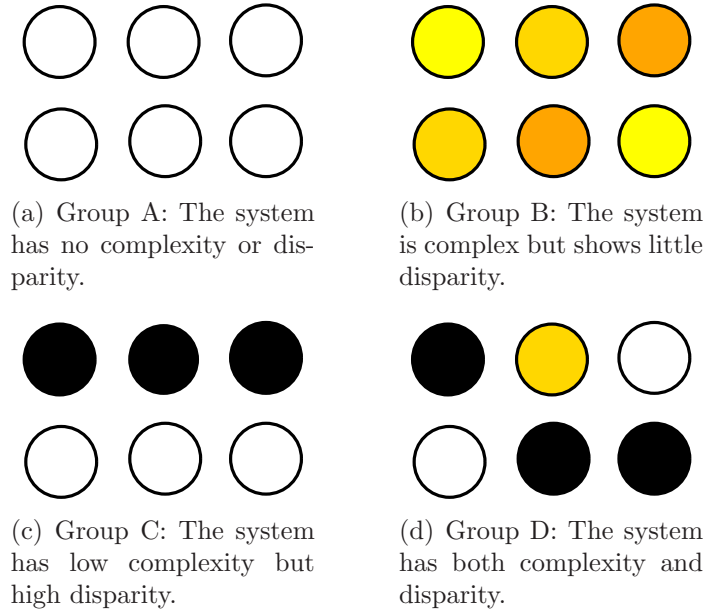


Figure 4: Different groups of colored marbles used to illustrate the importance of having both complexity and disparity in a heterogeneous system.

$$HETEROGENEITY = COMPLEXITY \times DISPARITY. \quad (1)$$

2.2 How can Heterogeneity be Quantified?

The previous example helped illustrate that heterogeneity depends on the complexity and disparity of a system. Now, we will further elaborate on this relationship by concisely stating how those quantities can be computed, which allows us to quantify heterogeneity using (1). It should be noted that the formulation of heterogeneity up to now is generic and can be applied to many scenarios. However, we will purposely phrase the upcoming discussion in the context of multi-agent systems to easily transition into the next section where issues specific to multi-agent systems are addressed.

2.2.1 A New Measure of Heterogeneity

Heterogeneity in a multi-agent system usually tends to partition the agents into a finite number of groups with different capabilities. For example, an ecosystem may

consist of many organisms but they can be categorized into a smaller set of species. Moreover, manufacturers may create large quantities of UAVs for the military to use but there may only be a small number of production models. To capture this for multi-agent systems in a generic manner, assume that the system consists of a fixed number of agents and that each agent belongs to exactly one of M possible species. Let $p_i \in [0, 1]$ be the probability that a randomly chosen agent belongs to species i , for $i \in \mathcal{M} = \{1, \dots, M\}$, such that

$$\sum_{i=1}^M p_i = 1, \text{ and where } p = [p_1, \dots, p_M]^T$$

is the vector of probabilities that a randomly chosen agent will belong to each of the available species. Moreover, let

$$\mathcal{P}_M = \left\{ p \in \mathbb{R}^M \mid p_i \in [0, 1] \text{ for } i \in \mathcal{M}, \text{ and } \sum_{i=1}^M p_i = 1 \right\}$$

be the set of all probability distributions over M species.

Recall that the complexity of a system describes how well spread out the agents are amongst the available species. This measure of disorder is exactly captured by the entropy of the system. Entropy, which is used often in information theory, gives the expected number of bits needed to describe which species an agent belongs to when using an optimal coding scheme. The precise definition of entropy is stated below.

Definition 2.2.1.1. *Suppose that the M available species in a multi-agent system, as well as the probability distribution $p \in \mathcal{P}_M$ of agents belonging to each species, have both been established. The entropy¹, $E : \mathcal{P}_M \rightarrow \mathbb{R}_{\geq 0}$, of the multi-agent system is given by*

$$E(p) = - \sum_{i=1}^M p_i \log(p_i). \quad (2)$$

¹Traditionally, in information theory, entropy is denoted by $H(p)$. However, in this paper, the letter H will instead be reserved for heterogeneity.

Entropy is minimized when all the agents belong to the same species as seen in Figure 5(a), and is maximized when the agents are evenly distributed across all species as illustrated in Figure 5(b). Note that entropy is purely a function of the percentage of agents belonging to each species.

Next, we move on to quantifying a system’s disparity. As was described before, disparity is a measure of how different agents are from one another. To compute such a quantity, it is necessary to first establish a metric amongst the set of all species. Disparity can then be computed as the expected squared distance between two randomly drawn agents by using Rao’s quadratic entropy.

Definition 2.2.1.2. *Suppose the entropy of a multi-agent system is well defined, and a metric $d : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}_{\geq 0}$ is established between the species. Rao’s quadratic entropy, $Q : \mathcal{P}_M \rightarrow \mathbb{R}_{\geq 0}$, of the multi-agent system is given by*

$$Q(p) = \sum_{i=1}^M \sum_{j=1}^M p_i p_j d(i, j)^2. \quad (3)$$

Note that just like with entropy, Rao’s quadratic entropy is also minimized when all agents belong to the same species as shown in Figure 5(a). The population distribution which maximizes Rao’s quadratic entropy involves the agents being distributed only amongst the pairs of species which are the most distant from one another based on the metric d , as seen in Figure 5(c).

Having specified how to compute both the complexity and disparity of a system, its heterogeneity can then be quantified using the relationship described in (1).

Definition 2.2.1.3. *Suppose that the entropy $E(p)$ and Rao’s quadratic entropy $Q(p)$ of a multi-agent system are both well-defined. The heterogeneity, $H : \mathcal{P}_M \rightarrow \mathbb{R}_{\geq 0}$, of a multi-agent system is given by*

$$H(p) = E(p) Q(p). \quad (4)$$

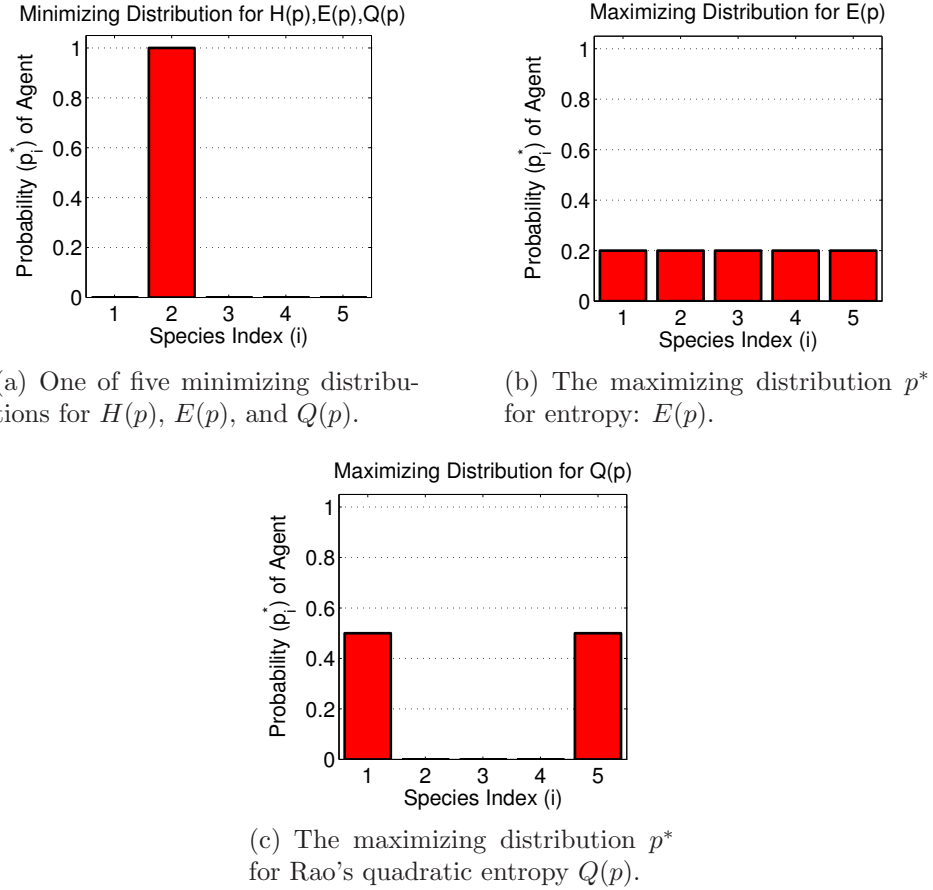


Figure 5: Plots showing the minimizing and maximizing distributions for entropy $E(p)$ and Rao's quadratic entropy $Q(p)$ in a multi-agent system with $M = 5$ uniform species, i.e., $d(i, j) = \alpha|i - j|$ for all $i, j \in \mathcal{M}$.

From the definition of H , we see that the distribution which minimizes heterogeneity is also whenever the entire population is concentrated in a single species, as shown in Figure 5(a). We state this observation in the lemma below.

Lemma 2.2.1.1. $H(p) \geq 0$ and is bounded for all probability distributions $p \in \mathcal{P}_M$. Furthermore, $H(p) = 0$ if and only if $p_k = 1$ for some $k \in \mathcal{M}$, and $p_i = 0$ for all $i \neq k$.

Proof. The boundedness and non-negativity of H follows from both E and Q being bounded and non-negative. Both E and Q equal zero if and only if the entire population is contained in one species, so the same holds for H . \square

2.2.2 Conservation of Species within Heterogeneity-Maximizing Populations

Recall that a major issue which prevented Rao's quadratic entropy from being used as a measure of biodiversity was that oftentimes, achieving the most diverse population required eliminating some species. We will now show that our heterogeneity measure does not suffer from this problem. Before showing this key result, however, we must first establish that a heterogeneity-maximizing distribution even exists.

Lemma 2.2.2.1. *There exists a probability distribution $p^* \in \mathcal{P}_M$ such that*

$$H(p^*) = \sup_{p \in \mathcal{P}_M} H(p).$$

Proof. H is a continuous function over the compact set \mathcal{P}_M . Therefore, a maximizing $p^* \in \mathcal{P}_M$ must exist. \square

We are now ready to show a key result: that no species need to be eliminated in the most heterogeneous population according to H , i.e., H promotes species variety.

Theorem 2.2.2.1. *Let $p^* \in \mathcal{P}_M$ be the probability distribution which maximizes H , then $p_i^* > 0$ for all $i \in \mathcal{M}$.*

Proof. The proof of this theorem will be shown using induction for when the system contains k species. In the base case when only $k = 1$ species is available, the only way to distribute the population is to have $p_1^* = 1 > 0$. For the inductive hypothesis, assume that for $k = M$ species, the population distribution which maximizes H assigns a nonzero portion of the population to each species. We will show that the same holds with $k = M + 1$ species.

First, note that with the $M + 1$ species available, the most heterogeneous distribution must have a nonzero population in at least M of the $M + 1$ species. The reason is that since H does not change if one of the vacant species is removed completely, the situation where $M + 1$ species are available but at least one species is empty can be treated as if there were only M species present. The inductive hypothesis then guarantees that all M of those remaining species will have nonzero populations.

Let $p \in \mathcal{P}_M$, where $p_1, \dots, p_M > 0$, be any probability distribution for an arbitrarily chosen subset of M species out of the $M + 1$ that are available. We will show that H will always increase when some of the population from each of the M species is distributed to the $M + 1$ th species. Hence, the most heterogeneous distribution when $k = M + 1$ must have nonzero population in all $M + 1$ species. Start by defining a probability distribution $q(\epsilon) \in \mathcal{P}_{M+1}$, which is parameterized by $\epsilon \geq 0$, where

$$q_i(\epsilon) = p_i - \epsilon \text{ and } q_{M+1} = M\epsilon,$$

for $i = 1, \dots, M$, such that ϵ represents how much of the population from the first M species is being transferred to the $M + 1$ th species. The heterogeneity of the new distribution is $H(q(\epsilon)) = E(q(\epsilon))Q(q(\epsilon))$, where

$$E(q(\epsilon)) = - \sum_{i=1}^M (p_i - \epsilon) \log(p_i - \epsilon) - M\epsilon \log(M\epsilon)$$

and

$$Q(q(\epsilon)) = \sum_{i=1}^M \sum_{j=1}^M (p_i - \epsilon)(p_j - \epsilon) d(i, j)^2 + 2M\epsilon \sum_{i=1}^M (p_i - \epsilon) d(i, M + 1)^2.$$

Notice that $H(q(\epsilon))$ is continuous for $\epsilon \in [0, \epsilon^*]$, where $\epsilon^* = \min\{\frac{1}{M}, p_1, \dots, p_M\}$ since $H(q(\epsilon))$ is only defined for $q(\epsilon) \in \mathcal{P}_{M+1}$. Taking the derivative with respect to ϵ gets

$$\frac{\partial H(q(\epsilon))}{\partial \epsilon} = \frac{\partial E(q(\epsilon))}{\partial \epsilon} Q(q(\epsilon)) + E(q(\epsilon)) \frac{\partial Q(q(\epsilon))}{\partial \epsilon},$$

where

$$\frac{\partial E(q(\epsilon))}{\partial \epsilon} = \sum_{i=1}^M \log(p_i - \epsilon) - M \log(M\epsilon),$$

and

$$\frac{\partial Q(q(\epsilon))}{\partial \epsilon} = \sum_{i=1}^M \sum_{j=1}^M (-p_i - p_j + 2\epsilon) d(i, j)^2 + 2M \sum_{i=1}^M (p_i - 2\epsilon) d(i, M+1)^2.$$

From this we see that $H(q(\epsilon))$ is continuously differentiable for $\epsilon \in (0, \epsilon^*)$.

Taking the right sided limit as ϵ goes to zero, we get that

$$\lim_{\epsilon \rightarrow 0^+} \frac{\partial H(q(\epsilon))}{\partial \epsilon} = \infty.$$

Therefore, there must be some $\hat{\epsilon} \in (0, \epsilon^*)$ for which $\frac{\partial H(q(\epsilon))}{\partial \epsilon}$ is positive for all $\epsilon \in (0, \hat{\epsilon})$ and hence, $H(p) < H(q(\epsilon))$. Since any such ϵ assigns a nonzero population to each of the $M+1$ species, heterogeneity always can be increased by distributing some of the population in the first M species to the $M+1$ th species. From this we see that when $k = M+1$, the heterogeneity-maximizing distribution involves having a nonzero population in each of the $M+1$ species. \square

2.2.3 Heterogeneity in Systems with Uniform Species

To gain better insight into what H considers the most heterogeneous population, we will temporarily focus on a special type of system. Suppose each species in a system is represented by its index, $1, \dots, M$, and the distance between two species is proportional to the absolute value of the difference between the two indices. We will refer to this situation collectively as the multi-agent system having uniform species.

Definition 2.2.3.1. A multi-agent system with uniform species consists of M available species, where the metric d between species is defined such that $d(i, j) = \alpha|i - j|$, for some constant $\alpha > 0$.

Although a system with uniform species is but a special case of the many types of systems that can exist, it allows us to easily visualize population distributions in a simple scenario where not all species are equidistant. The intuition gained from analyzing this special case can then be applied to estimate what the maximally heterogeneous population distribution looks like in a general system.

We will start by showing that the most heterogeneous population distribution in a system with uniform species obeys a symmetry property.

Lemma 2.2.3.1. If p^* is the heterogeneity-maximizing distribution in a multi-agent system with uniform species, then it satisfies the symmetry property: $p_i^* = p_{M+1-i}^*$, for all $i \in \mathcal{M}$.

Proof. We will show that whenever a probability distribution p does not satisfy the symmetry property, it is always possible to construct a new distribution $q \in \mathcal{P}_M$, where

$$q_i = q_{M+1-i} = \frac{1}{2}(p_i + p_{M+1-i}), \text{ for all } i \in \mathcal{M},$$

such that q satisfies the symmetry property and $H(q) > H(p)$.

First, we show that $E(q) > E(p)$. Substituting q into the definition of E and performing some simplifications gives

$$E(q) = -\sum_{i=1}^M q_i \log(q_i) = -\sum_{i=1}^M p_i \log(q_i)$$

Suppose in the above expression we fix p and are allowed to choose $q \in \mathcal{P}_M$ to minimize $E(q)$. Using Lagrange multipliers to minimize $E(q)$ with the constraint that $g(q) = \sum_{i=1}^M q_i - 1 = 0$, it is simple to show that $\arg \min_{q \in \mathcal{P}_M} E(q) = p$. However, since $q \neq p$, we then conclude that $E(q) > E(p)$.

Next, we will show that $Q(q) \geq Q(p)$. Substituting q into the definition of Q and performing some simplifications, while keeping in mind that

$$d(i, j)^2 = d(M + 1 - i, M + 1 - j)^2$$

because the system has uniform species, gives

$$Q(q) = \frac{1}{2}Q(p) + \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M p_i p_{M+1-j} d(i, j)^2.$$

Computing the difference between $Q(q)$ and $Q(p)$ and performing some simplifications results in

$$\begin{aligned} Q(q) - Q(p) &= \alpha \sum_{i=1}^M \sum_{j=1}^M p_i p_j (i + j - M - 1)^2 - \alpha \sum_{i=1}^M \sum_{j=1}^M p_i p_j (i - j)^2 \\ &= \alpha (4\epsilon_p^2 - 4(M + 1)\epsilon_p + (M + 1)^2) \\ &= \alpha (2\epsilon_p - M - 1)^2 \geq 0, \end{aligned}$$

where $\epsilon_p = \sum_{j=1}^M p_j j$ is the expected species index of a randomly drawn agent. Because we have shown that $E(q) > E(p)$ and $Q(q) \geq Q(p)$, it then follows that $H(q) > H(p)$. \square

Building on the symmetry result in the previous lemma, we will now derive a key relationship between the percentages of agents in each species within the most heterogeneous distribution.

Lemma 2.2.3.2. *If p^* is the probability distribution which maximizes H in a multi-agent system with uniform species, then $p_i^* > p_k^* \iff i^2 - k^2 > (i - k)(M + 1)$.*

Proof. The problem of finding the heterogeneity-maximizing distribution can be posed as a constrained maximization problem, where the goal is to maximize $H(p)$ with respect to the constraints: $g(p) = 0$ and $h_i(p) \geq 0$, for all $i \in \mathcal{M}$, such that

$$g(p) = \sum_{i=1}^M p_i - 1 \text{ and } h_i(p) = p_i,$$

for all $i \in \mathcal{M}$. The resulting Lagrange function is given by

$$\Lambda(p, \lambda, \mu) = H(p) + \lambda g(p) + \sum_{i=1}^M \mu_i h_i(p),$$

where $\lambda \neq 0$ and $\mu_i \geq 0$, for $i \in \mathcal{M}$, are Lagrange multipliers. However, since it was shown in Theorem 2.2.2.1 that $p_i^* > 0$ for all $i \in \mathcal{M}$, we know that all of the inequality constraints are inactive and so $\mu_i = 0$, for all $i \in \mathcal{M}$. Therefore, we can simplify the Lagrange function to:

$$\Lambda(p, \lambda) = H(p) + \lambda g(p).$$

The optimality conditions state that

$$\frac{\partial \Lambda(p, \lambda)}{\partial p_k} = \frac{\partial H(p)}{\partial p_k} + \lambda = 0, \text{ for all } k \in \mathcal{M}.$$

Calculating the partial derivative of $H(p)$ and substituting it into the previous expression, we get that for all $k \in \mathcal{M}$:

$$-(\log(p_k) + 1) Q(p) + 2E(p) \sum_{j=1}^M p_j d(j, k)^2 = -\lambda.$$

We will use the following shorthand notation for the summation in the expression above:

$$X_k^2 = \sum_{j=1}^M p_j d(j, k)^2.$$

Since λ is constant, the following holds for any $i, k \in \mathcal{M}$:

$$-\log(p_i) Q(p) + 2E(p) X_i^2 = -\log(p_k) Q(p) + 2E(p) X_k^2$$

Grouping the quantities together, we get the following optimality condition for each pair of species $i, k \in \mathcal{M}$:

$$\frac{2E(p)}{Q(p)} (X_k^2 - X_i^2) = \log\left(\frac{p_k}{p_i}\right).$$

By Lemma 2.2.1.1, when $M \geq 2$, the heterogeneity-maximizing distribution causes $H(p) > 0$, and so $E(p) > 0$ and $Q(p) > 0$. Therefore, for the above expression to hold, it must be that

$$\text{sgn}(p_i - p_k) = \text{sgn}(X_i^2 - X_k^2), \tag{5}$$

for all $i, k \in \mathcal{M}$.

We will now calculate the term $X_i^2 - X_k^2$ for the special case of when the multi-agent system has uniform species.

$$\begin{aligned}
X_i^2 - X_k^2 &= \sum_{j=1}^M p_j (d(i, j)^2 - d(j, k)^2) \\
&= \alpha \sum_{j=1}^M p_j ((i - j)^2 - (j - k)^2) \\
&= \alpha \sum_{j=1}^M p_j (i^2 - k^2 - 2j(i - k)) \\
&= \alpha (i^2 - k^2 - 2(i - k) \epsilon_p).
\end{aligned}$$

However, because the heterogeneity maximizing probability distribution is symmetric by Lemma 2.2.3.1, we have that $\epsilon_p = \frac{M+1}{2}$. Substituting it back into the previous expression gives

$$X_i^2 - X_k^2 = \alpha (i^2 - k^2 - (i - k)(M + 1)). \quad (6)$$

Combining this result with the expression in (5) gives

$$\text{sgn}(p_i - p_k) = \text{sgn}(i^2 - k^2 - (i - k)(M + 1)),$$

from which we conclude that

$$p_i > p_k \iff i^2 - k^2 > (i - k)(M + 1).$$

□

The previous lemma describes which species has more of the population distributed to it than others in the most heterogeneous population. Combined with the symmetry result of Lemma 2.2.3.1, they give a concise statement about the shape of the most heterogeneous population distribution in a system with uniform species.

Theorem 2.2.3.1. *Let p^* be the probability distribution which maximizes the heterogeneity measure H in a multi-agent system with uniform species, then $p_i^* > p_{i+1}^*$ for all $i < \frac{M}{2}$, and $p_k^* = p_{M+1-k}^*$ for all $k \in \mathcal{M}$.*

Proof. Letting $k = i + 1$ in Lemma 2.2.3.2 gives

$$p_i^* > p_{i+1}^* \iff i^2 - (i + 1)^2 > (i - (i + 1))(M + 1),$$

where upon simplifying the inequality on the right side,

$$p_i^* > p_{i+1}^* \iff \frac{M}{2} > i.$$

That shows the first part of the theorem. The second part of the theorem is simply a restatement of Lemma 2.2.3.1. \square

Theorems 2.2.2.1 and 2.2.3.1 together paint a picture of what the most heterogeneous population distribution looks like with uniform species. With this information, we can now compare the heterogeneity measure H with entropy and Rao's quadratic entropy. Figure 6 shows the heterogeneity maximizing distribution in a multi-agent system with uniform species, consisting of $M = 5$ species total. Notice that consistent with Theorem 2.2.2.1, the maximally heterogeneous distribution assigns agents to each species, unlike with Rao's quadratic entropy as seen in Figure 5(c). Furthermore, the heterogeneity-maximizing distribution does not assign the same number of agents to each species in all situations, unlike with entropy as seen in Figure 5(b). Instead, because the differences between species are taken into account, more agents are assigned to the species whose indices are closer to the two extremes: 1 and 5, as described by Theorem 2.2.3.1. Thus, the most heterogeneous distribution retains some of the population in all species, and strikes a balance amongst the system's complexity and disparity.

2.3 How Heterogeneous is a Multi-Agent System?

The computation of H , or more specifically Q , in (4) requires that a metric d be established amongst the species. However, this choice may seem rather arbitrary when dealing with agents in a general setting. For example, what is the distance

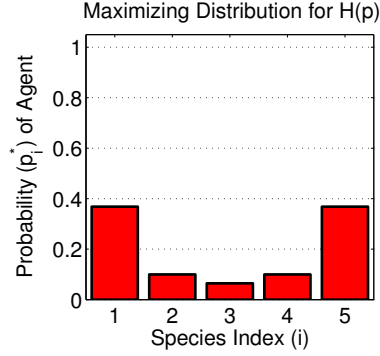


Figure 6: Plot showing the heterogeneity maximizing distribution for H in a multi-agent system with uniform species, where $M = 5$. Notice that as stated in Theorem 2.2.2.1, all species have a nonzero portion of the total population assigned to it. Moreover, the shape of the distribution agrees with the description provided by Theorem 2.2.3.1.

between a robot dog that can only wag its tail and a robot parrot that can only sing? Should the two be compared by their weight, size, vocal abilities, physical abilities, color, or something else? Clearly, the correct answer to this question depends on the task at hand. Most realistic multi-agent missions, however, will consist of multiple tasks that need to be accomplished by agents in the system instead of just one. To capture this objective, we will use the notion of a task-space to describes a set of tasks that are of interest. Such a set allows us to describe and compare the capabilities of agents from different species on a common ground that is relevant to the intended application of the agents.

Definition 2.3.0.2. A task-space², (T, γ) , is a pair containing a non-empty countable set T of tasks, and a weight function $\gamma : T \rightarrow \mathbb{R}_{>0}$ for each task such that

$$\sum_{t \in T} \gamma(t) = 1.$$

The purpose of the weight function in the definition is to place greater emphasis on certain tasks over others. This could be for various reasons, such as if one task

²We realize that this definition of task-space is somewhat different from the ones found in existing robotics literature.

occurs more frequently than another. Accompanying a task-space is a task-map which describes what tasks each species of agents can accomplish.

Definition 2.3.0.3. A task-map, $\omega : \mathcal{M} \rightarrow 2^T$, associated with a multi-agent system and its task-space is a mapping from species index to a set of tasks.

These definitions establish a common ground for which the capabilities of different agent species can be described and compared on. To perform such a comparison, we present the notion of a functional distance as a measure of the difference in two species' capabilities with respect to a task space.

Definition 2.3.0.4. The functional distance between two species $i, j \in \mathcal{M}$, with respect to a task-space (T, γ) and task-map ω , is given by $\delta(T, \gamma, \omega) : \mathcal{M} \times \mathcal{M} \rightarrow [0, 1]$, where

$$\delta(T, \gamma, \omega)(i, j) = \frac{\sum_{t \in (\omega(i) \cup \omega(j)) \setminus (\omega(i) \cap \omega(j))} \gamma(t)}{\sum_{u \in \omega(i) \cup \omega(j)} \gamma(u)}. \quad (7)$$

Suppose $\gamma(t)$ describes the percentage of the mission which requires an agent to perform task t . Then the functional distance is simply the ratio of the number of tasks which only one of the two agents can accomplish, to the number of tasks that either can do. When two agents are capable of the exact same tasks, the functional distance is 0. Moreover, when two agents perform sets of tasks which are disjoint, the functional distance is 1. Having established a distance measure with respect to a task space, we can compute the heterogeneity of a multi-agent system with respect to that space. Since this measure is based on the agents' ability to carry out tasks specific to the task-space, we will refer to it as the system's functional heterogeneity.

Definition 2.3.0.5. The functional heterogeneity $\mathcal{H}(p, T, \gamma, \omega)$ of a multi-agent system with respect to a task-space (T, γ) and task-map ω , is the heterogeneity $H(p)$ of the system using the functional distance $\delta(T, \gamma, \omega)$ as the species metric.

2.3.1 Examples: Heterogeneous Search-and-Rescue

We will now illustrate how to compute the functional heterogeneity for a multi-agent system through an example of a heterogeneous search-and-rescue effort taking place in an aquatic environment.

Example 2.3.1.1. *Consider a mission where UAVs and AUVs are dispatched to search a body of water for a target, as was illustrated earlier in Figure 3. Areas where the target could be located are categorized as shallow water, deep water, and underwater cave. The task space is therefore given by (T, γ) , where*

$$T = \{Shallow, Deep, Cave\},$$

and

$$\gamma(Shallow) = 0.5, \gamma(Deep) = 0.3, \gamma(Cave) = 0.2,$$

tells how much of the search space belongs to each category. Each UAV is equipped with cameras that allows it to look at both shallow and deep water. Meanwhile, the AUVs can only propel themselves in deep water, but can venture into underwater caves. Letting UAVs and AUVs be the two agent species, indexed 1 and 2 respectively, the task maps are:

$$\omega(1) = \{Shallow, Deep\} \text{ and } \omega(2) = \{Deep, Cave\}.$$

Suppose a total of 2 UAVs and 2 AUVs (which we will refer to collectively as Team A) are dispatched for the mission, i.e., $p_A = [0.5 \ 0.5]^T$. The complexity of team A is:

$$E(p_A) = -2(0.5) \log(0.5) = 0.6931.$$

The functional distance between species 1 and 2 is given by

$$\begin{aligned} \delta(T, \gamma, \omega)(1, 1) &= \delta(T, \gamma, \omega)(2, 2) = 0, \\ \delta(T, \gamma, \omega)(1, 2) &= \delta(T, \gamma, \omega)(2, 1) = 0.7. \end{aligned}$$

Computing the disparity of the system yields

$$Q(p_A) = 2(0.5)(0.5)(0.7)^2 = 0.2450.$$

Putting the two together allows us to compute the functional heterogeneity of Team A with respect to our task space as:

$$\mathcal{H}(p_A, T, \gamma, \omega) = E(p_A)Q(p_A) = 0.1698.$$

Thus, this example illustrates that through the use of a common task space, the heterogeneity metric can be applied to multi-agent systems. Lower-level details that describe how an agent goes about doing a task, which may be different when agents are heterogeneous, are abstracted out and instead the focus is on the higher-level capabilities of the agents. To illustrate this point further, the heterogeneity of another multi-agent system will now be computed and compared to that from the previous example.

Example 2.3.1.2. *We will now consider what happens to the functional heterogeneity of the system when a new type of agent is added to the system: a speedboat driven by humans. Let this new species have index 3. Since the speedboat's movement is confined to the surface of the water, humans on-board can only see what is in shallow water. Therefore,*

$$\omega(3) = \{Shallow\}.$$

This new team involving 2 UAVs, 2 AUVs, and 1 speedboat will be referred to collectively as Team B, where $p_B = [0.4 \ 0.4 \ 0.2]^T$. The complexity of Team B is given by

$$E(p_B) = -2(0.4) \log(0.4) - (0.2) \log(0.2) = 1.0549.$$

Notice that the introduction of an agent from a new species immediately raises the complexity of the system.

The functional distance of the UAVs and AUVs relative to the speedboat is given by

$$\begin{aligned}\delta(T, \gamma, \omega)(1, 3) &= \delta(T, \gamma, \omega)(3, 1) = 0.375, \\ \delta(T, \gamma, \omega)(2, 3) &= \delta(T, \gamma, \omega)(3, 2) = 1, \\ \delta(T, \gamma, \omega)(3, 3) &= 0.\end{aligned}$$

The disparity associated with the new system is therefore

$$Q(p_B) = 2(0.4)(0.4)(0.7)^2 + 2(0.4)(0.2)(0.375)^2 + 2(0.4)(0.2)(1)^2 = 0.3393.$$

Putting the two together, we see that the resulting functional heterogeneity of Team B with respect to our task space is

$$\mathcal{H}(p_B, T, \gamma, \omega) = E(p_B)Q(p_B) = 0.3579.$$

Therefore, by adding a single speedboat to the team of UAVs and AUVs, the functional heterogeneity of the system has increased with respect to the task of searching the body of water shown in Figure 3.

To summarize, this chapter on network design focused on first developing a basic understanding of heterogeneity in multi-agent systems. By using the concept of complexity and disparity, a unifying metric for quantifying heterogeneity was proposed that was shown to overcome the problems that existing diversity metrics suffer from. Moreover, the metric was shown to be directly applicable to describing the heterogeneity of a multi-agent system in terms of each agent's individual ability to accomplish tasks within a finite discrete set of tasks. Now, armed with an understanding of heterogeneity, the next chapter will explore its implications on the expressiveness of multi-agent systems.

CHAPTER III

NETWORK DESIGN PART 2: EXPRESSIVENESS

Having developed an understanding of heterogeneity in Chapter 2, this second chapter on network design will now explore how the expressiveness of a multi-agent system is influenced by its heterogeneity (or lack thereof). Recall that in the previous chapter, heterogeneity in multi-agent systems was defined in terms of the capabilities of the agents, i.e., the kinds of tasks each agent could do in the common task-space. Consider now, the scenario where a set of tasks located at target points τ_1, \dots, τ_N need to be accomplished and the expressiveness of a network is defined by how closely one can control the agents to those target points so as to perform the tasks. Since it is not feasible to directly take control of every agent in the network, a single-leader paradigm will be considered where the state of a single agent is directly controlled and is used to affect the states of the remaining agents. Expressiveness of the network can then be phrased in terms of the controllability properties of the single-leader network.

In this chapter, however, we will investigate what happens to a system's controllability when homogeneity is present in the network. In particular, the scenario where agents have the same capabilities so that any agent i can be assigned to any target point τ_j will be considered. Viewed from the perspective of traditional controllability, this corresponds to the task of moving the agents to any permutation of the target points. Since the majority of network topologies yield single-leader networks which are not completely controllable, exploiting the homogeneity of agents in this case can conceivably allow for target points which are not reachable to be moved closer to a system's reachable subspace through permuting its labels. However, to fully analyze such a control strategy requires a new way to understand and view controllability

since it has changed from a traditional point-to-point property to now a point-to-set property, where the set corresponds to all permutations of the target.

This chapter is organized as follows: Section 3.1 first mathematically describes the dynamics of the single-leader networks whose controllability properties will be analyzed. Then, Section 3.2 will reveal the interesting controllability properties associated with single-leader networks when the agents have homogeneous capabilities. It is shown that the computational complexity of finding an optimal permutation of a target point so as to move it closest to a homogeneous system's reachable subspace is NP-hard. Therefore, Section 3.3 looks deeper into when solving for the optimal solution can yield target points that avoid worst-case scenarios such as being orthogonal to the reachable subspace. Finally, Section 3.4 presents methods to find suboptimal permutations of the target point for when solving for the optimal solution is computationally infeasible.

3.1 System Dynamics

To begin, consider a team of $N + 1$ agents, numbered $1, \dots, N + 1$, with positions $x_i \in \mathbb{R}^n$, for $i = 1, \dots, N + 1$, respectively. Let the information flow amongst agents in the network be represented by a static undirected graph $G = (V, E)$, where $V = \{v_1, \dots, v_{N+1}\}$ and $(v_i, v_j) \in E$ if and only if information flows between agents i and j . The neighbor set $N_i = \{j \mid (v_i, v_j) \in E\}$ represents the index set of all agents that share an edge with agent i in G .

Suppose the agents form a single-leader network where all followers execute a nearest neighbor averaging rule based on sensed relative information in order to maintain cohesion, while the leader's position is the external input u . Without loss of generality, assume the $N + 1$ th agent is the leader while agents $1, \dots, N$ are followers. The

agents' dynamics are:

$$\begin{cases} \dot{x}_i &= - \sum_{j \in N_i} (x_i - x_j), \text{ for } i = 1, \dots, N, \\ x_{N+1} &= u. \end{cases} \quad (8)$$

It should be noted here that the agent dynamics are completely driven by sensed information and that no communication is being considered. This is a constraint imposed on the network based on the practical need to conserve energy for each agent, thereby elongating the operational time the agents, since inter-agent communication generally consumes much more energy than inter-agent sensing.

The adjacency matrix of G is the $(N + 1) \times (N + 1)$ symmetric matrix A where $A_{i,j}$, the element in the i th row and j th column, is given by

$$A_{i,j} = \begin{cases} 1 & , \text{ if } (v_i, v_j) \in E, \\ 0 & , \text{ otherwise.} \end{cases} \quad (9)$$

The degree matrix of the graph G is a $(N + 1) \times (N + 1)$ diagonal matrix Δ , where

$$\Delta_{i,j} = \begin{cases} |N_i| & , \text{ if } i = j, \\ 0 & , \text{ otherwise.} \end{cases} \quad (10)$$

Finally, the graph Laplacian matrix L is given by

$$L = \Delta - A, \quad (11)$$

which can be decomposed into blocks

$$L = \begin{bmatrix} L_f & \ell \\ \ell^T & \xi \end{bmatrix}, \quad (12)$$

where the dimension of L_f is $N \times N$, ℓ is $N \times 1$, and $\xi \in \mathbb{R}$.

Let $x = [x_1^T, \dots, x_N^T]^T \in \mathbb{R}^{Nn}$ be the concatenated positions of all follower agents, where $x_j = [x_{j,1}, \dots, x_{j,n}]^T \in \mathbb{R}^n$, for $j = 1, \dots, N$. Define $d_i : \mathbb{R}^{Nn} \rightarrow \mathbb{R}^N$, for $i = 1, \dots, n$, as a function that returns the positions of the N follower agents along

the i th dimension, i.e., $d_i(x) = [x_{1,i}, \dots, x_{N,i}]^T$. The dynamics of the follower agents' positions along the i th dimension are given by

$$d_i(\dot{x}) = -L_f d_i(x) - \ell u_i, \quad (13)$$

where u_i is the i th element of u . Since the dynamics along each dimension are decoupled, the dynamics of x can be written using the Kronecker product, as the linear system

$$\dot{x} = -(L_f \otimes I_n)x - (\ell \otimes I_n)u, \quad (14)$$

where I_n is the $n \times n$ identity matrix, e.g., [72].

3.2 Controllability in Homogeneous Single-Leader Networks

To understand how closely homogeneous agents in a single-leader network can be controlled to a set of target points, a new way of understanding controllability will be developed in this section. However, before that can be done, it is necessary to first consider the controllability properties of single-leader networks where the agents are not homogeneous.

3.2.1 Controllability of Single-Leader Networks

In a single-leader network, the dynamics of the follower agents along each dimension are decoupled and given by the linear system (13). Treating each dimension separately, the reachable subspace is given by the range space of the controllability Grammian Γ , where

$$\Gamma = \begin{bmatrix} -\ell & L_f \ell & \dots & (-L_f)^{N-1} (-\ell) \end{bmatrix}. \quad (15)$$

The reachable subspace of a single-leader network was found in [68] to have an interesting interpretation involving the graph topology. Before stating this result, we must first review some definitions from [40, 68, 72].

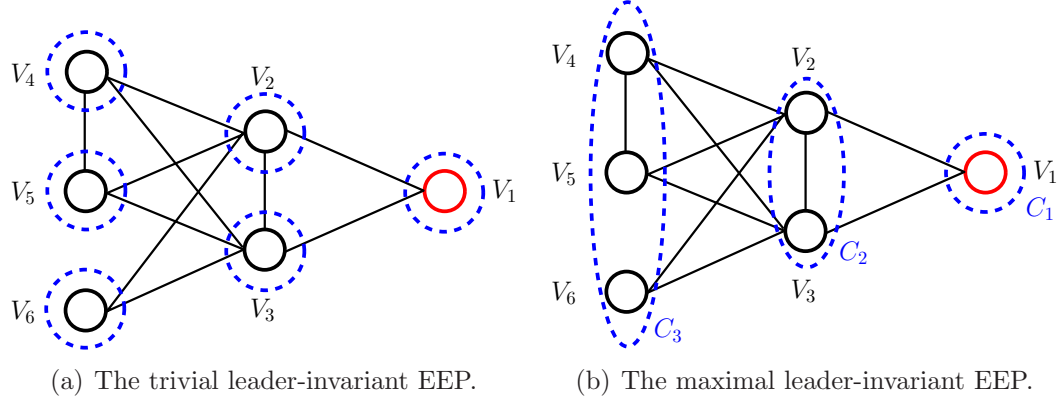


Figure 7: Two examples of leader-invariant EEPs of a single-leader network, where V_1 is the vertex for the leader agent. (a) shows the trivial leader-invariant EEP, while (b) gives the maximal leader-invariant EEP. Since the two partitions are different, the network is not completely controllable.

Definition 3.2.1.1. Given a vertex set V , let $\Pi = \{C_1, \dots, C_M\}$ be a partition of V , where $C_i \subseteq V$ for $i = 1, \dots, M$, $C_1 \cup \dots \cup C_M = V$, and $C_i \cap C_j = \emptyset$ when $i \neq j$. We will call each C_i a cell.

Definition 3.2.1.2. Given a vertex v and a cell C , the node-to-cell degree gives the number of vertices in cell C that share an edge with v , and is given by $\text{deg}(v, C) = \text{card}(\{v' \in C \mid (v, v') \in E\})$.

For example, in Figure 7(b), C_1, C_2, C_3 are cells that partition the vertices in the network and $\text{deg}(v_2, C_3) = 3$.

Definition 3.2.1.3. An external equitable partition (EEP) is a partition Π such that $\forall C \in \Pi$, then $v \in C$ and $v' \in C \Rightarrow \text{deg}(v, C') = \text{deg}(v', C') \forall C' \in \Pi - \{C\}$.

Definition 3.2.1.4. An EEP is leader-invariant if the vertex corresponding to the leader agent belongs to its own cell.

Definition 3.2.1.5. A leader-invariant EEP is maximal if it has the fewest number of cells in any leader-invariant EEP.

For example, Figure 7(a) is a leader-invariant EEP, while Figure 7(b) is a maximal leader-invariant EEP.

With these definitions, we now state a result relating the controllability of a single-leader network to its maximal leader-invariant EEP. In [68] it was shown that cells of the maximal leader-invariant EEP give information as to which groups of follower agents cannot be controlled independently of one another. That result is stated again below for easy reference.

Theorem 3.2.1.1. *[68] Assume a single-leader network has a network topology with a maximal leader-invariant EEP of k^* cells, numbered $1, \dots, k^*$, that do not contain the leader agent. The range space of the controllability Grammian for (13), the follower agent dynamics along any dimension, is given by:*

$$R(\Gamma) = \text{span} \{w_1, \dots, w_k\}, \quad (16)$$

for some $1 \leq k \leq k^*$, and where $w_i \in \mathbb{R}^N$. Moreover, letting $w_{i,j}$ represent the j th element of w_i , it must be that

1. $w_{i,j} \in \{0, 1\}$,
2. $w_{i,a} = 1 \Rightarrow w_{i,b} = 1$ for all v_b that belongs in the same cell as v_a ,
3. $\sum_{i=1}^k w_i = \mathbf{1}$, where $\mathbf{1}$ is the vector of all 1's.

The theorem states that follower agents which are located within the same cell of the maximal leader-invariant EEP will asymptotically approach each other as they move according to the dynamics (14). Therefore, instead of being able to control each agent's position independently, only the centroid of agents within each cell can be controlled. However, sometimes the centroids of some cells cannot be controlled independently of one another as well, resulting in all the agents of those cells asymptotically approaching each other. This is why $1 \leq k \leq k^*$.

It should be noted here that the the lack of complete controllability stems from the symmetry within the single-leader network as revealed using the maximal leader-invariant EEP. This symmetry is an artifact of the idealized assumption where each

agent is executing the exact same control law and can be oftentimes be alleviated by having each agents' control signal use a different gain value. Such a break in the symmetry can either be artificially imposed onto the system through the usage of a randomly generated gain for each agent, or can be also caused by noise in sensor readings and slight differences in actuator dynamics for each agent. The focus of controllability analysis on the ideal case is because of the insight offered for the non-ideal cases, since it is likely that an unreachable state in the ideal case may be reachable in the non-ideal case but will require extraordinary amounts of control effort to do so.

Moreover, the analysis to be performed will focus on single-leader networks which satisfy the following key assumption.

Assumption 3.2.1.1. *In this chapter, we restrict our attention to single-leader networks in which the reachable subspace $R(\Gamma)$ is completely determined by agents' membership within the cells of the maximal leader-invariant EEP. In other words, we assume that $k = k^*$ in Theorem 3.2.1.1. Therefore, the reachable subspace $R(\Gamma) = \text{span}\{w_1, \dots, w_k\}$ is such that*

$$w_{i,j} = \begin{cases} 1 & , \text{ if } v_j \in \text{cell } i, \\ 0 & , \text{ otherwise.} \end{cases} \quad (17)$$

Although this may seem like a major assumption, it is not overly restrictive since most network topologies for single-leader networks do indeed satisfy this assumption. Whenever the assumption holds true, a single-leader network is completely controllable and can reach any target point only when the maximal leader-invariant EEP is trivial, i.e., each follower agent is contained within its own cell. Referring back to Figure 7, we see that the trivial leader-invariant EEP is not the same as the maximal leader-invariant EEP and so the network is not completely controllable. For general single-leader networks, the results which are based on the above assumption can be viewed as an upper bound on the limits of the system's controllability.

3.2.2 Optimally Reachable Target Points

Recall that agents which belong in the same maximal leader-invariant EEP will asymptotically approach one another over time and cannot be controlled independently. To maintain the focus of the analysis on long-term control objectives where such transient effects are no longer of concern, we make the following key assumption which states that agents belonging in the same cell start and stay together always.

Assumption 3.2.2.1. *All agent positions are initially zero.*

With zero initial conditions on x , a target point of follower agents $x_T \in \mathbb{R}^{Nn}$ is reachable if and only if $d_i(x_T) \in R(\Gamma)$, for $i = 1, \dots, n$. Depending on the network topology, the system of follower agents is not always completely controllable and so may not be able to reach a target point perfectly. Therefore, for a given x_T , the best that can be done is to drive the system to the optimal reachable target point $x^*(x_T)$, which minimizes

$$J(x_T, x) = \|x_T - x\|^2 = \sum_{i=1}^n \|d_i(x_T) - d_i(x)\|^2, \quad (18)$$

such that $d_i(x) \in R(\Gamma)$, for $i = 1, \dots, n$.

Theorem 3.2.2.1. *For a given x_T , the optimal reachable $x^*(x_T)$, where $d_i(x^*(x_T)) \in R(\Gamma)$, for $i = 1, \dots, n$, that minimizes (18) is*

$$x^*(x_T) = (WW^T \otimes I_n) x_T, \quad (19)$$

where

$$W = \begin{bmatrix} \frac{w_1}{\|w_1\|} & \cdots & \frac{w_k}{\|w_k\|} \end{bmatrix}, \quad (20)$$

and w_1, \dots, w_k are as given in (16).

Proof. Minimizing $\|x_T - x\|^2$ is equivalent to minimizing $\|d_i(x_T) - d_i(x)\|$ individually for each i because the dynamics along each dimension are decoupled. The

Hilbert Projection Theorem says that the optimal reachable $d_i(x^*(x_T)) \in R(\Gamma)$ that minimizes $\|d_i(x_T) - d_i(x)\|$ is the projection of $d_i(x_T)$ onto the subspace $R(\Gamma)$. The reachable subspace $R(\Gamma)$ is spanned by vectors w_1, \dots, w_k as given in (16). Therefore, the optimal choice of $d_i(x)$ is given by

$$d_i(x^*(x_T)) = \sum_{j=1}^k \frac{w_j^T d_i(x_T)}{\|w_j\|^2} w_j. \quad (21)$$

For W as defined in (20), (21) can be rewritten as

$$d_i(x^*(x_T)) = WW^T d_i(x_T).$$

Since this holds for all i , $x^*(x_T)$ is written as (19). □

The expression determined for $x^*(x_T)$ has an interesting and intuitive interpretation that will be useful later. Define $g_i : \mathbb{R}^{Nn} \rightarrow \mathbb{R}^n$, for $i = 1, \dots, N$, as a function that returns the n dimensional coordinates of the i th agent, i.e., $g_i(x) = x_i$. Further, define $m : \{1, \dots, N\} \rightarrow \{1, \dots, k\}$ as a function that takes in an index of a follower agent and returns the index of the cell it belongs to in the maximal leader-invariant EEP. Let m^{-1} be the inverse image function that takes in a cell number and returns a set containing the indices of the follower agents that belong to that cell.

Corollary 3.2.2.1. *For a given x_T and corresponding $x^*(x_T)$ that minimizes (18),*

$$g_i(x^*(x_T)) = \frac{1}{|m^{-1}(m(i))|} \sum_{j \in m^{-1}(m(i))} g_j(x_T). \quad (22)$$

In other words, agents in cell j of $x^(x_T)$ are all located at the centroid of the target positions in cell j of x_T .*

Proof. From the definition of vectors w_j in (17), the expression for $d_i(x^*(x_T))$ in (21) can be interpreted. The numerator of each summand $w_j^T d_i(x_T)$ is the sum along the i th dimension of all target positions in cell j . That quantity is divided by $\|w_j\|^2$, which is the number of agents in cell j , so the result is the centroid along the i th

dimension of all target positions in cell j of x_T . Finally, that value is multiplied to w_j , thereby assigning it to the i th dimensional component of all agents positions in cell j of $x^*(x_T)$. Since this holds for along all dimensions $i = 1, \dots, n$, agents in cell j of $x^*(x_T)$ are all located at the centroid of the target positions in cell j of x_T . \square

With an expression for $x^*(x_T)$, we can compute the minimum cost associated with any given x_T .

Corollary 3.2.2.2. *For a given x_T and corresponding $x^*(x_T)$, the minimum cost $J^*(x_T) = J(x_T, x^*(x_T))$ is*

$$J^*(x_T) = x_T^T (I_{Nn} - WW^T \otimes I_n) x_T. \quad (23)$$

Proof. Plugging in the expression (19) for x^* into the cost (18), expanding the norm-squared, and noticing that the term $I_{Nn} - WW^T \otimes I_n$ is symmetric results in

$$J^*(x_T) = x_T^T (I_{Nn} - WW^T \otimes I_n)^2 x_T.$$

Expanding the squared term and using the fact that the columns of W are orthonormal yields (23). \square

3.2.3 Homogeneous Networks

Equation (23) represents the cost associated with the closest that a particular single-leader network can reach a target point x_T . Notice that x_T represents the specification to have each agent i be located at $g_i(x_T)$, for $i = 1, \dots, N$. However, in a network of homogeneous agents, the roles of agents are interchangeable and so it makes no difference if instead we ask agent i to go to $g_j(x_T)$ and agent j to go to $g_i(x_T)$. In fact, any permutation of the agent indices in x_T to some $(P \otimes I_n) x_T$, where P is a permutation matrix, ends up specifying the same target configuration if all we care about is the presence of an agent at each of the target positions. However, the new target point may be “more reachable” in the sense that $J^*((P \otimes I_n) x_T) < J^*(x_T)$.

Example 3.2.3.1. Consider a single-leader network with scalar (1D) agent positions and $N = 3$ follower agents as illustrated in Figure 8(a), where agents 1 and 2 are in cell 1 and agent 3 is in cell 2 of the maximal leader-invariant EEP of the network. The range space of the controllability Grammian is thus

$$R(\Gamma) = \text{span} \{w_1, w_2\} = \left\{ \left[\begin{array}{c} 1 \\ 1 \\ 0 \end{array} \right], \left[\begin{array}{c} 0 \\ 0 \\ 1 \end{array} \right] \right\}.$$

Let

$$x_T = \begin{bmatrix} 1 \\ 9 \\ 10 \end{bmatrix} \quad \text{and} \quad P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix},$$

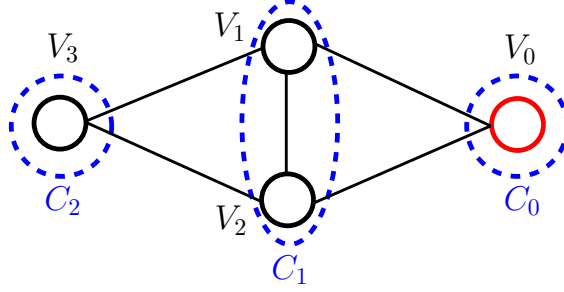
then $J^*(x_T) = 32$, while $J^*(Px_T) = 0.5$. Therefore, as illustrated in Figures 8(b) and 8(c), Px_T , the permuted target point, is more reachable than the original target point x_T .

The previous example showed that different permutations of a target point may be at different distances from the system's reachable subspace. Finding the optimal permutation which brings a target closest to the reachable subspace, therefore, requires solving the following problem:

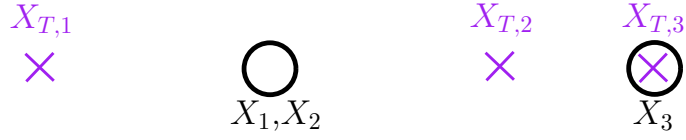
Problem 3.2.3.1. Let \mathcal{P} be the set of all $N \times N$ permutation matrices. Given a single-leader network and target point of follower agents x_T , find P^* such that

$$P^* = \arg \min_{P \in \mathcal{P}} J^*((P \otimes I_n) x_T). \quad (24)$$

Calculating P^* can be viewed as finding the optimal specification of a target configuration for the follower agents. However, a more intuitive interpretation of finding P^* can be found by treating it as a constrained clustering problem on the N target positions $g_1(x_T), \dots, g_N(x_T)$. To do so, it is first necessary to review some clustering terminology.



(a) The single-leader network used in Example 3.2.3.1, where the leader agent's vertex is V_0 .



(b) The closest the follower agents in the network (circles) can reach target point x_T (X's).



(c) The closest the follower agents in the network (circles) can reach the permuted target points Px_T (X's).

Figure 8: The topology of the single-leader network in Example 3.2.3.1 is given in (a). (b) shows the closest the follower agents can reach $x_T = [1 \ 9 \ 10]^T$, while (c) shows the closest the follower agents can reach $Px_T = [9 \ 10 \ 1]^T$. Notice that the Px_T results in an error less than x_T and so Px_T is the better specification of the target configuration.

Definition 3.2.3.1. A multiset is a collection of objects in which order is ignored, but where multiplicity is significant.

For example, $M_1 = \{1, 3, 4\}$, $M_2 = \{1, 3, 4, 4\}$, and $M_3 = \{1, 4, 3, 4\}$ are all multisets. $M_2 = M_3$, but $M_1 \neq M_2$ and $M_1 \neq M_3$. Also, $|M_1| = 3$, while $|M_2| = |M_3| = 4$.

Definition 3.2.3.2. Given a multiset S , a clustering of S is a partitioning of the elements of S into multisets c_1, \dots, c_k .

Now, let S be a multiset of agent positions. Within each cluster c_i , define the distortion measure of that cluster as

$$D(c_i) = \sum_{z \in c_i} \|z - \theta(c_i)\|^2, \quad (25)$$

where $\theta(c_i)$ is the centroid of all positions in c_i . Define the cost of a clustering as the total distortion measure, given by

$$H(c_1, \dots, c_k) = \sum_{i=1}^k D(c_i). \quad (26)$$

Problem 3.2.3.2. The Euclidean minimum sum-of-squares clustering problem is to find a clustering c_1^*, \dots, c_k^* , given a multiset of positions S , so as to minimize (26).

Theorem 3.2.3.1. Suppose a single-leader network has a maximal leader-invariant EEP of exactly k cells containing follower agents, numbered $1, \dots, k$. Finding the optimal permutation P^* for a target x_T in Problem 3.2.3.1 is equivalent to solving Problem 3.2.3.2 under Assumption 3.2.1.1 for the multiset of target positions, $S = \{g_1(x_T), \dots, g_N(x_T)\}$, with the constraint that $|c_i| = |m^{-1}(i)|$, the number of agents in cell i , for $i = 1, \dots, k$.

Proof. Given a permutation matrix P , let $p : \{1, \dots, N\} \rightarrow \{1, \dots, N\}$ take in an agent index and return the permuted index such that, for $j = 1, \dots, N$, $g_j(x_T) =$

$g_{p(j)}((P \otimes I_n) x_T)$. Let $c_i = \{g_j((P \otimes I_n) x_T) \mid m(j) = i\}$, for $i = 1, \dots, k$, be a clustering of $S = \{g_1(x_T), \dots, g_N(x_T)\}$, where target positions in $(P \otimes I_n) x_T$ with indices in cell i are assigned to c_i .

Notice that $|c_i| = |m^{-1}(i)|$, the number of agents in each cell i , for $i = 1, \dots, k$. Considering different permutations of agent indices for the target point x_T is equivalent to considering different cell assignments of the target positions, which is equivalent to considering clusterings c_1, \dots, c_k of S . The cost (23) associated with a chosen permutation P of target positions can be rewritten using (18) and (22) as

$$\begin{aligned}
& J^*((P \otimes I_n) x_T) \\
&= \sum_{i=1}^n \|d_i((P \otimes I_n) x_T) - d_i(x^*((P \otimes I_n) x_T))\|^2 \\
&= \sum_{i=1}^N \|g_i((P \otimes I_n) x_T) - g_i(x^*((P \otimes I_n) x_T))\|^2 \\
&= \sum_{i=1}^N \|g_i((P \otimes I_n) x_T) - \theta(c_{m(i)})\|^2 \\
&= \sum_{i=1}^k \sum_{j \mid m(j)=i} \|g_j((P \otimes I_n) x_T) - \theta(c_i)\|^2 \\
&= \sum_{i=1}^k \sum_{z \in c_i} \|z - \theta(c_i)\|^2 = H(c_1, \dots, c_k),
\end{aligned}$$

which shows that the cost is equivalent to (26).

Given the P^* that solves Problem 3.2.3.1, an optimal clustering c_1^*, \dots, c_k^* that solves Problem 3.2.3.2 under the constraint that $|c_i| = |m^{-1}(i)|$, for $i = 1, \dots, k$, can be computed by the polynomial-time algorithm:

Let c_1^*, \dots, c_k^* be empty multisets;
for $i = 1, \dots, N$ **do**
 | Add $g_i((P^* \otimes I_n) x_T)$ to $c_{m(i)}^*$;
end

Alternatively, given an optimal clustering c_1^*, \dots, c_k^* , the matrix P^* can be computed by the polynomial time algorithm:

Let $Q = R = \{1, \dots, N\}$, and $P^* = 0$ ($N \times N$ matrix);

for $i = 1 \dots, k$ **do**

for each $z \in c_i^*$ **do**

 Find any $j \in Q$ such that $g_j(x_T) = z$;

 Remove j from Q ;

 Find a $b \in m^{-1}(i)$ such that $b \in R$;

 Remove b from R ;

 Set the element $P_{b,j}^*$ to 1;

end

end

Thus, finding P^* in Problem 3.2.3.1 under Assumption 3.2.1.1 is equivalent to finding an optimal clustering c_1^*, \dots, c_k^* for $S = \{g_1(x_T), \dots, g_N(x_T)\}$, that minimizes (26) subject to $|c_i|$ equaling the number of agents in cell i , for $i = 1, \dots, k$. \square

Viewing the problem of finding the optimal permutation as a size-constrained version of the Euclidean minimum sum-of-squares clustering problem is useful because it allows us find the computational complexity associated with the task.

Theorem 3.2.3.2. *The problem of finding the optimal permutation matrix P^* in Problem 3.2.3.1 under Assumption 3.2.1.1 is NP-hard.*

Proof. It was shown in [3] that the Euclidean minimum sum-of-squares clustering problem described in Problem 3.2.3.2 is NP-hard by using a reduction from the DENSEST CUT problem for the case of $k = 2$ clusters. Using almost the same procedure, we will show that the optimization version of the MAX BISECTION problem, which was shown in [87] to be NP-hard, reduces to the size-constrained Euclidean minimum sum-of-squares problem for $k = 2$ clusters.

Let $G = (V, E)$ be an undirected graph. Define B_1, B_2 as a partition of V such that $|B_1| = |B_2| = \frac{N}{2}$, where N is assumed to be even. The MAX BISECTION

problem is to find B_1^* and B_2^* so as to maximize $|\mathcal{E}(B_1, B_2)|$, where $\mathcal{E}(B_1, B_2) = \{(v_i, v_j) \in E \mid v_i \in B_1 \text{ and } v_j \in B_2\}$.

Arbitrarily number and orient the edges in E as $e_1, \dots, e_{|E|}$ so that each e_i is an ordered pair of vertices. Define the incidence matrix \mathcal{I} as a $N \times |E|$ matrix such that for each $e_k = (v_i, v_j) \in E$, $\mathcal{I}_{i,k} = -1$ and $\mathcal{I}_{j,k} = 1$. Have $x_1, \dots, x_N \in \mathbb{R}^{|E|}$ be such that x_i^T equals the i th row of \mathcal{I} . Define the multiset $S = \{x_1, \dots, x_N\}$. Have c_1, c_2 be two clusters that partition S subject to the size constraint $|c_1| = |c_2| = \frac{N}{2}$. Let B_1 and B_2 be a partition of V , where $B_i = \{v_j \mid x_j \in c_i\}$, for $i = 1, 2$.

Let the function $\phi_j : \mathbb{R}^{|E|} \rightarrow \mathbb{R}$ take in a vector and return the j th element of its argument. Computing the total distortion of the cluster as in (26), we have

$$\begin{aligned} H(c_1, c_2) &= \sum_{i=1}^2 \sum_{z \in c_i} \|z - \theta(c_i)\|^2 \\ &= \sum_{j=1}^{|E|} \sum_{i=1}^2 \sum_{z \in c_i} (\phi_j(z) - \phi_j(\theta(c_i)))^2 \end{aligned}$$

If $e_j \in \mathcal{E}(B_1, B_2)$, then either $\phi_j(z)$ equals 1 for exactly one $z \in c_1$ and equals -1 for exactly one $z \in c_2$ with all others equaling 0 and thus $\phi_j(\theta(c_1)) = \frac{2}{N}$ and $\phi_j(\theta(c_2)) = -\frac{2}{N}$, or the same statements above but with c_1 and c_2 switched. Furthermore, if $e_j \notin \mathcal{E}(B_1, B_2)$, then $\phi_j(\theta(c_1)) = \phi_j(\theta(c_2)) = 0$. Using these properties:

$$\begin{aligned} H(c_1, c_2) &= \sum_{e \in \mathcal{E}(B_1, B_2)} \sum_{i=1}^2 \left(\left(\frac{N}{2} - 1 \right) \left(\frac{2}{N} \right)^2 + \left(1 - \frac{2}{N} \right)^2 \right) + \sum_{e \notin \mathcal{E}(B_1, B_2)} 2 \\ &= 2 \left(1 - \frac{2}{N} \right) |\mathcal{E}(B_1, B_2)| + 2 (|\mathcal{E}(B_1, B_1)| + |\mathcal{E}(B_2, B_2)|) \\ &= 2|E| - \frac{4}{N} |\mathcal{E}(B_1, B_2)|. \end{aligned}$$

Choice of B_1^* and B_2^* , or equivalently the choice of c_1^* and c_2^* , that minimizes $H(c_1, c_2)$ also maximizes $|\mathcal{E}(B_1, B_2)|$, since $|E|$ and N are constant. Therefore, the NP-hard MAX BISECTION problem reduces to the size-constrained Euclidean minimum sum-of-squares problem, which is itself equivalent to finding P^* in Problem

3.2.3.1 under Assumption 3.2.1.1. This brings us to the conclusion that finding P^* is also NP-hard. \square

3.3 When to Solve for Optimal Permutations

Having established that finding the optimal permutation in Problem 3.2.3.1 under Assumption 3.2.1.1 is NP-hard, the next step is to identify when it is worthwhile to perform such a computation. In particular, for a given single-leader network, we wish to characterize how far any optimally permuted target point will be from the system's reachable subspace. With that information, an appropriate strategy for finding an optimal or suboptimal permutation can then be applied accordingly. We will first address this problem for when agent positions are scalar ($1D$), and then later consider the general case where agents have nD position coordinates.

3.3.1 Optimal Permutations with $1D$ Agents

To determine how close a single-leader follower network with $1D$ agent positions can get to any optimally permuted target point, consider the following problem.

Problem 3.3.1.1. *Given a single-leader network with $N \geq 2$ follower agents, $1D$ agent positions, and reachable subspace $R(\Gamma)$, find*

$$M_1(\Gamma) = \max_{\|x\|=1} \left\{ \min_{P \in \mathcal{P}} \|\Pi_{R(\Gamma)^\perp}(Px)\|^2 \right\}, \quad (27)$$

where $\Pi_a(b)$ is the projection of vector b onto a .

Note that $M_1(\Gamma) \in [0, 1]$, where $M_1(\Gamma) = 1$ corresponds to the worst case scenario when target points exist that remain orthogonal to the system's reachable subspace no matter how they are permuted. In the case when $\text{rank}(\Gamma) = 1$, i.e., when all follower agents cannot be moved independently of one another, the following shows that such a worst case scenario occurs.

Theorem 3.3.1.1. *The solution to Problem 3.3.1.1 for when $\text{rank}(\Gamma) = 1$ is given by $M_1(\Gamma) = 1$.*

Proof. Since $\text{rank}(\Gamma) = 1$, then $R(\Gamma) = \text{span}\{\mathbf{1}\}$. Let $x \in \mathbb{R}^N$ be such that $\|x\| = 1$ and $x^T \mathbf{1} = 0$. Then for any $P \in \mathcal{P}$,

$$(Px)^T \mathbf{1} = x^T P^T \mathbf{1} = x^T (P^{-1} \mathbf{1}) = x^T \mathbf{1} = 0.$$

Thus, $\|\Pi_{R(\Gamma)^\perp}(Px)\|^2 = 1$ for any $P \in \mathcal{P}$ and so $M_1(\Gamma) = 1$. \square

Such a result should not be too surprising since if $\text{rank}(\Gamma) = 1$, then all follower agents are confined to move together. Under such a scenario, the closest that a single-leader network can meet a target point is to have all agents go to the centroid of the targets, which is invariant to permutations. Fortunately, however, this is not true for networks where the reachable subspace has higher rank, i.e., $1 < \text{rank}(\Gamma) \leq N$. To show this, the following result must first be established.

Lemma 3.3.1.1. *For a single-leader network as described in Problem 3.3.1.1 with $1 < \text{rank}(\Gamma) \leq N$, $Px \perp R(\Gamma)$ for all $P \in \mathcal{P} \iff x = 0$.*

Proof. First, we prove the necessary condition (\Leftarrow). If $x = 0$, then $Px = 0$ for all $P \in \mathcal{P}$ and 0 is orthogonal to all possible subspaces.

To show the sufficient condition (\Rightarrow), two cases need to be considered. First, if $\text{rank}(\Gamma) = N$, then $R(\Gamma) = \mathbb{R}^N$. In this case, the only vector x which is orthogonal to $R(\Gamma)$ is $x = 0$. Now consider the case when $1 < \text{rank}(\Gamma) < N$, which can only occur when $N \geq 3$. Here, $R(\Gamma)$ must be spanned by a basis vector $v \in \mathbb{R}^N$ that consists of 0's and at least 2 but less than N 1's. If $Px \perp R(\Gamma)$ for all $P \in \mathcal{P}$, then certainly $v^T (Px) = 0$ for all $P \in \mathcal{P}$ as well. Given $i, j \in \{1, \dots, N\}$, it is then possible to find permutation matrices $P_{ijA} \in \mathcal{P}$ and $P_{ijB} \in \mathcal{P}$ where

$$v^T (P_{ijA}x) = v^T (P_{ijB}x) = 0 \Rightarrow x_i = x_j.$$

Since this can be done for any $i, j \in \{1, \dots, N\}$, it must be that $x = \alpha \mathbf{1}$ for some $\alpha \in \mathbb{R}$. However, since $v^T x = 0$, it must be that $\alpha = 0$ and thus $x = 0$. \square

With this result, an important bound on $M_1(\Gamma)$ for $1 < \text{rank}(\Gamma) \leq N$ can be made.

Theorem 3.3.1.2. *The solution to Problem 3.3.1.1 for when $1 < \text{rank}(\Gamma) \leq N$ is bounded by $M_1(\Gamma) < 1$.*

Proof. Since when calculating $M_1(\Gamma)$ we only consider x such that $\|x\| = 1$, it is not the case that $x = 0$. Therefore, by Lemma 3.3.1.1, we see that for each target point x there exists a $P^*(x) \in \mathcal{P}$ such that $P^*(x)x \notin R(\Gamma)$ and so $\|\Pi_{R(\Gamma)^\perp}(P^*(x)x)\|^2 < 1$. Because this is true for all target points x such that $\|x\| = 1$, we conclude that $M_1(\Gamma) < 1$. \square

Theorems 3.3.1.1 and 3.3.1.2 give insight into when it is worthwhile to solve the NP-hard Problem 3.2.3.1 for an optimal permutation. With the exception of when $\text{rank}(\Gamma) = 1$, optimal permutations were shown to help single-leader networks avoid having target points be orthogonal to the system's reachable subspace.

3.3.2 Optimal Permutations with nD Agents

Having given some insight as to when computing the optimal permutation is worthwhile for a system with $1D$ agent positions, we now consider the general case of agents with nD positions. Following a similar approach as in Problem 3.3.1.1, we wish to bound how far any optimally permuted target point would be from the system's reachable subspace. When dealing with agents that have nD positions, the dynamics along each dimension have the same reachable subspace. Thus, the generalization of $M_1(\Gamma)$ to $M_n(\Gamma)$ should bound the furthest that an optimally permuted target point can be from the reachable subspace along any dimension. The generalized version of Problem 3.3.2.1 is the following:

Problem 3.3.2.1. *Given a single-leader network with $N \geq 2$ follower agents, nD*

agent positions, and reachable subspace $R(\Gamma)$ along each dimension, find

$$M_n(\Gamma) = \max_{\substack{\|d_j(x)\| = 1 \\ j = 1, \dots, n}} \left\{ \min_{P \in \mathcal{P}} \left\{ \max_{k=1, \dots, n} \|\Pi_{R(\Gamma)^\perp}(Pd_k(x))\|^2 \right\} \right\}. \quad (28)$$

Like its single-dimensional predecessor, Problem 3.3.2.1 is also not trivial to solve. The main difference lies in how the increased dimensionality of agent positions affects the distance that optimally permuted target points are from the system's reachable subspace. We will now show that for networks where $1 \leq \text{rank}(\Gamma) < N$, the worst case scenario of having a target be orthogonal to the reachable subspace becomes unavoidable as the dimension of agents increases.

Theorem 3.3.2.1. *The solution to Problem 3.3.2.1 for when $n \geq N!$ and $1 \leq \text{rank}(\Gamma) < N$ is $M_n(\Gamma) = 1$.*

Proof. To prove this theorem, we will construct a target point $x^* \in \mathbb{R}^{Nn}$ that will cause $M_n(\Gamma) = 1$. Assume that as stated in the theorem, $n \geq N!$ and $1 \leq \text{rank}(\Gamma) < N$. Since the system is not completely controllable, let x^* be such that $d_1(x^*) \perp R(\Gamma)$. Moreover, let $d_2(x^*), \dots, d_{N!}(x^*)$ represent all $N! - 1$ other permutations of $d_1(x^*)$. In that case, no matter which $P \in \mathcal{P}$ is used, there is always a $k \in \{1, \dots, N!\}$ such that $Pd_k(x^*) \perp R(\Gamma)$ and so $\|\Pi_{R(\Gamma)^\perp}(Pd_k(x^*))\|^2 = 1$. The ability to construct such a target point x^* means that $M_n(\Gamma) = 1$. \square

3.4 Finding Suboptimal Permutations

In the previous section, the problem of bounding the distance that an optimally permuted target point can be from the system's reachable subspace was presented. The solution serves to help determine when it is worthwhile to solve the NP-hard Problem 3.2.3.1 for a particular single-leader network. In this section, methods are presented to solve for suboptimal permutations in the event that the computational cost of solving the NP-hard problem outweighs the benefits.

3.4.1 Heuristic and Approximation Algorithms

A commonly used method for finding locally optimal solutions to the Euclidean sum of squares problem is the k-means algorithm (e.g., [62]). However, Theorem 3.2.3.1 adds equality constraints on the size of individual clusters. In [16], a constrained k-means clustering algorithm is proposed that finds locally optimal clusterings which minimize (26), where the minimum size of individual cluster can be specified. Equality constraints on the cluster sizes are imposed when minimum cluster sizes are chosen to sum to N .

3.4.2 Special Case: 1D Networks

For general single-leader networks, finding P^* in Problem 3.2.3.1 involves considering at most $N!$ possible permutation matrices, or equivalently $N!$ clusterings by Theorem 3.2.3.1. However, in the special case of 1D networks where the maximal leader-invariant EEP consists of k cells with followers, only $k!$ clusterings need to be considered by exploiting a special property.

Definition 3.4.2.1. *In a clustering c_1, \dots, c_k of a multiset S of 1D points, a cluster c_i is compact if $\nexists x_{i1}, x_{i2} \in c_i$ and $\nexists x_j \in c_j$ such that $x_{i1} < x_j < x_{i2}$, $\forall j \neq i$.*

Lemma 3.4.2.1. *The optimal clustering c_1^*, \dots, c_k^* of a multiset S of 1D points, which minimizes (26) fixed $|c_i|$ for $i = 1, \dots, k$, involves only compact clusters.*

Proof. We start by showing that elements in every non-compact clustering can always be reassigned to decrease (26) without changing the cluster sizes. Assume c_1, \dots, c_k are not all compact, then $\exists x_{a1}, x_{a2} \in c_a$ and $x_b \in c_b$ such that $x_{a1} < x_b < x_{a2}$, for some c_a and c_b where $a \neq b$. Furthermore, define

$$H_o(c_1, \dots, c_k, m_1, \dots, m_k) = \sum_{i=1}^k \sum_{z \in c_i} (z - m_i)^2,$$

where $H(c_1, \dots, c_k) \leq H_o(c_1, \dots, c_k, m_1, \dots, m_k)$ with equality when $m_i = \theta(c_i)$, for $i = 1, \dots, k$. The total distortion of the clustering can be rewritten as

$$\begin{aligned} H(c_1, \dots, c_k) &= H_o(c_1, \dots, c_k, \theta(c_1), \dots, \theta(c_k)) \\ &= Q - 2R(c_a, c_b, \theta(c_a), \theta(c_b)), \end{aligned}$$

where

$$Q = \sum_{i=1}^N x_i^2 + \sum_{i=1}^k |c_i| \theta(c_i)^2 - 2 \sum_{i=1, i \neq a, b}^k \theta(c_i) \sum_{z \in c_i} z,$$

and

$$R(c_a, c_b, m_a, m_b) = m_a \sum_{z \in c_a} z + m_b \sum_{z \in c_b} z.$$

If $\theta(c_a) \geq \theta(c_b)$, assign \hat{c}_a the $|c_a|$ largest elements of $c_a \cup c_b$, while giving \hat{c}_b the remaining elements. Otherwise, if $\theta(c_a) < \theta(c_b)$, then let \hat{c}_a have the $|c_a|$ smallest elements of $c_a \cup c_b$, while \hat{c}_b gets the rest. Furthermore, define $\hat{c}_i = c_i \forall i \neq a, b$. Notice that $|\hat{c}_i| = |c_i|$, for $i=1, \dots, k$. Then since after the reassignment, $\theta(\hat{c}_a) \neq \theta(c_a)$ and $\theta(\hat{c}_b) \neq \theta(c_b)$,

$$\begin{aligned} H(c_1, \dots, c_k) &= Q - 2R(c_a, c_b, \theta(c_a), \theta(c_b)) \\ &\geq Q - 2R(\hat{c}_a, \hat{c}_b, \theta(c_a), \theta(c_b)) \\ &= H_o(\hat{c}_1, \dots, \hat{c}_k, \theta(c_1), \dots, \theta(c_k)) \\ &> H(\hat{c}_1, \dots, \hat{c}_k). \end{aligned}$$

Therefore, whenever a clustering c_1, \dots, c_k is not all compact, it is possible to obtain a new clustering $\hat{c}_1, \dots, \hat{c}_k$ with a lower total distortion. Since there are only a finite number of ways to cluster points in S , an optimal cluster must exist and it must involve only compact clusters. \square

The previous lemma established that the optimal clustering must indeed be compact. Such a result is utilized in the following theorem to reduce the number of clusterings that need to be searched.

Theorem 3.4.2.1. *Finding c_1^*, \dots, c_k^* to minimize (26) for a 1D single-leader network requires considering at most $k!$ clusterings.*

Proof. For 1D points, only the ordering of the k compact clusterings matter in finding c_1^*, \dots, c_k^* . Thus, at most only $k!$ clusterings need to be considered. \square

To summarize, this chapter presented and explored the problem of multi-agent system controllability when agent homogeneity was taken into account explicitly. Finding the optimal permutation matrix that moved a target point closest to a multi-agent system's reachable subspace was shown to be NP-hard. Analytical results were presented to give insight into when solving for the optimal permutation using computationally expensive methods could help avoid the worst-case scenario of having a target point be orthogonal to the system's reachable subspace. Moreover, some heuristic approaches were presented for the case when finding the optimal solution was computationally infeasible. Combined with the results from the previous chapter on understanding heterogeneity, these two chapters gave the designer of a multi-agent network a better understanding of the relationship between heterogeneity and expressiveness.

CHAPTER IV

GENERATING DECENTRALIZED CONTROLLERS

Moving onwards from network design, the next step is to come up with a tool that helps generate decentralized coordination strategies that agents can then use to accomplish a desired task with. In general, the process of designing decentralized control laws is rather difficult. As mentioned in Section 1.2.3, existing approaches to designing decentralized controllers can be categorized into either a top-down or bottom-up approach. In the top-down approach, controllers are developed so as to optimize some global objective function and the resulting controllers are revealed to have a structure which supports decentralization. In the bottom-up approach, a decentralized control law is proposed and an analysis is performed to show the resulting global properties of the system when agents execute that controller. Oftentimes, however, it may be necessary to design a control law which allows for agents to move in state trajectories so as to optimize some global objective function, but at the same time impose a form on the types of decentralized control laws which can be used. In particular, there may be limitations on the information flow amongst agents or computational capabilities of agents that need to be enforced explicitly.

An example of such a scenario can be the need to design decentralized control laws to enable a team of humans in a marching band to track a globally-specified trajectory. Here, the global objective is to minimize the tracking error between the state trajectory and the desired trajectory. Moreover, to make the control law potentially implementable by humans, one can base each agent's motion off of sensed relative displacements from only a few other agents in the network and compute the velocity command of each agent as simply a scaled and rotated version of the sensed relative

displacement vector.

This chapter will present research on bridging the top-down and bottom-up approaches to decentralized controller design so as to generate decentralized controllers that can explicitly incorporate both a global objective function and an imposed structure on the decentralized control law. In particular, the optimal decentralization algorithm from [123, 125] will be presented. Optimal decentralization provides a way to generate a sequence of decentralized controllers for agents to track a desired multi-agent motion that has been defined on the agent-trajectory level. Given parameterized constraints which describe what constitutes a decentralized controller for the system, parameters are computed so as to make agents track the desired trajectories the best. Note that the reason why a sequence of controllers is used, as opposed to a single static controller, is to allow for the network to track complex motions such as that shown in Figure 1.

This chapter is organized as follows: First, Section 4.1 will present how to optimize parameterized modes in a switched autonomous system, so as to minimize some cost functional. Then, Section 4.2 will apply the derived results to the context of multi-agent networks in an example application involving tracking a desired multi-agent motion. Finally, Section 4.3 will present simulation results that showcase agents tracking a complex drumline-inspired multi-agent dance using decentralized controllers generated from the algorithm.

4.1 Optimizing Parameterized Modes

Consider a dynamical system that evolves as a switched autonomous system starting at time $t = 0$, and ending at time $t = T$, with m modes and $m - 1$ switching times. Each of the modes' dynamics are given by the function f but are parameterized by different scalar parameters γ_k , for each mode $k = 1, \dots, m$. The switching times are

$\tau_1, \dots, \tau_{m-1}$ satisfying

$$0 = \tau_0 \leq \tau_1 \leq \dots \leq \tau_{m-1} \leq \tau_m = T, \quad (29)$$

with the k th mode occurring in the time interval $[\tau_{k-1}, \tau_k)$. Letting

$$\Gamma = [\gamma_1, \dots, \gamma_m]^T \quad (30)$$

contain the parameters for all modes, the dynamics of the system are given by

$$\dot{x} = F(x, \Gamma, t) \quad (31)$$

with

$$F(x, \Gamma, t) = f(x, \gamma_k) \quad \forall t \in [\tau_{k-1}, \tau_k). \quad (32)$$

The objective is to choose Γ so that the resulting state trajectory minimizes the generalized cost functional

$$J = \int_0^T H(x(t)) dt. \quad (33)$$

Theorem 4.1.0.2. *The optimality condition for each γ_k in (32) with respect to cost (33) is*

$$\frac{\partial J}{\partial \gamma_k} = \int_{\tau_{k-1}}^{\tau_k} p^T(\tau) \frac{\partial f}{\partial \gamma_k}(x(\tau), \gamma_k) d\tau = 0, \quad (34)$$

where p is the costate with dynamics

$$\dot{p} = - \left(\frac{\partial F}{\partial x} \right)^T p - \left(\frac{\partial H}{\partial x} \right)^T \quad (35)$$

with boundary condition

$$p(T) = 0. \quad (36)$$

Proof. Perturbing the parameter γ_k that defines the k th mode, the dynamics of the perturbed system are

$$\dot{x} + \Delta \dot{x} = \begin{cases} \vdots \\ f(x, \gamma_{k-1}), & t \in [\tau_{k-2}, \tau_{k-1}) \\ f(x + \Delta x, \gamma_k + \Delta \gamma_k), & t \in [\tau_{k-1}, \tau_k) \\ f(x + \Delta x, \gamma_{k+1}), & t \in [\tau_k, \tau_{k+1}) \\ \vdots \end{cases}$$

with $x(0) + \Delta x(0) = x(0) = x_0$. The dynamics of the deviation Δx are given by a first-order approximation as

$$\Delta \dot{x} = \begin{cases} 0, & t \in [0, \tau_{k-1}) \\ \frac{\partial f}{\partial x} \Delta x + \frac{\partial f}{\partial \gamma_k} \Delta \gamma_k, & t \in [\tau_{k-1}, \tau_k) \\ \frac{\partial f}{\partial x} \Delta x, & t \in [\tau_k, \tau_{k+1}) \\ \vdots & \end{cases}$$

where $\Delta x(0) = \Delta x(\tau_{k-1}) = 0$. Letting $\Phi(\cdot)$ be the state transition matrix for the Δx linear system, the dynamics can be rewritten as

$$\Delta x = \begin{cases} 0, & t \in [0, \tau_{k-1}) \\ \int_{\tau_{k-1}}^t \Phi(t, \tau) \frac{\partial f}{\partial \gamma_k}(\tau) \Delta \gamma_k d\tau, & t \in [\tau_{k-1}, \tau_k) \\ \Phi(t, \tau_k) \int_{\tau_{k-1}}^{\tau_k} \Phi(\tau_k, \tau) \frac{\partial f}{\partial \gamma_k}(\tau) \Delta \gamma_k d\tau, & t \in [\tau_k, T]. \end{cases}$$

To derive the optimality conditions, it is necessary to calculate

$$J(\gamma_k + \Delta \gamma_k) - J(\gamma_k) = \frac{\partial J}{\partial \gamma_k} \Delta \gamma_k = \Delta J_{\gamma_k}, \quad (37)$$

where

$$\Delta J_{\gamma_k} = \int_0^T (H(x + \Delta x) - H(x)) dt \approx \int_0^T \frac{\partial H}{\partial x} \Delta x dt$$

simplifies to

$$\Delta J_{\gamma_k} = \int_{\tau_{k-1}}^{\tau_k} \left(\int_{\tau}^T \frac{\partial H}{\partial x}(t) \Phi(t, \tau) dt \right) \frac{\partial f}{\partial \gamma_k}(\tau) d\tau \Delta \gamma_k.$$

Defining the costate as

$$p^T(\tau) = \int_{\tau}^T \frac{\partial H}{\partial x}(t) \Phi(t, \tau) dt, \quad (38)$$

the expression for ΔJ_{γ_k} can be rewritten as

$$\Delta J_{\gamma_k} = \int_{\tau_{k-1}}^{\tau_k} p^T(\tau) \frac{\partial f}{\partial \gamma_k}(\tau) d\tau \Delta \gamma_k.$$

Seeing that the previous equation matches the form in (37), the optimality condition for γ_k are given by

$$\frac{\partial J}{\partial \gamma_k} = \int_{\tau_{k-1}}^{\tau_k} p^T(\tau) \frac{\partial f}{\partial \gamma_k}(\tau, \gamma_k) d\tau. \quad (39)$$

Now, it is necessary to derive an expression for the dynamics and boundary conditions of the costate. Taking the time-derivative of the costate defined in (38) results in

$$\dot{p}^T(\tau) = \frac{\partial}{\partial \tau} \left(- \int_T^\tau \frac{\partial H}{\partial x}(t) \Phi(t, \tau) dt \right).$$

Applying the chain rule and substituting in the state transition matrix property $\frac{\partial}{\partial \tau} \Phi(t, \tau) = -\Phi(t, \tau) \frac{\partial F}{\partial x}(\tau)$ gets

$$\dot{p}^T(\tau) = -\frac{\partial H}{\partial x}(\tau) - \int_\tau^T \frac{\partial H}{\partial x}(t) \Phi(t, \tau) \frac{\partial F}{\partial x}(\tau) dt.$$

Moving terms out of the integral gives

$$\dot{p}^T(\tau) = - \left(\int_\tau^T \frac{\partial H}{\partial x}(t) \Phi(t, \tau) dt \right) \frac{\partial F}{\partial x}(\tau) - \frac{\partial H}{\partial x}(\tau),$$

where by reapplying the definition of the costate in (38), the costate is shown to evolve as

$$\dot{p} = - \left(\frac{\partial F}{\partial x} \right)^T p - \left(\frac{\partial H}{\partial x} \right)^T. \quad (40)$$

The boundary condition for the costate is found by letting $\tau = T$ in (38), which yields

$$p(T) = 0. \quad (41)$$

□

The optimality conditions for switching times in a switched autonomous system were derived in [34]. They are restated here, for the sake of easy reference:

Theorem 4.1.0.3. *The optimality condition with respect to cost (33) for switching times in a switched autonomous system, where mode k has dynamics f parameterized by γ_k , is*

$$\frac{\partial J}{\partial \tau_k} = p^T(\tau_k) (f(x(\tau_k), \gamma_k) - f(x(\tau_k), \gamma_{k+1})) = 0. \quad (42)$$

The costate dynamics are the same as (35), with associated boundary conditions (36).

4.2 *Application: Tracking Motions using Multi-Agent Networks*

The derived results from the previous section can be applied to generate a sequence of decentralized controllers for use on a multi-agent network. To illustrate this, an example application will now be provided where agents must use a specific set of parameterized decentralized control laws to best track a multi-agent motion. It should be noted that the agent dynamics, decentralized control laws, and cost functional (i.e., for tracking) which will be used are for the sake of this particular example application. However, the tools developed in the previous section are general enough to be applicable to a much wider range of multi-agent systems and tasks.

Suppose that a system of N agents are located in a plane, where the position of the i th agent is given by its state $x_i \in \mathbb{R}^2$, for $i = 1, \dots, N$. Let the information flow amongst agents in the network be described using a static directed graph $G = (\mathcal{N}, E)$. The vertex set $\mathcal{N} = \{1, \dots, N\}$ is a set of N nodes, corresponding to the N agents. The edge set E is defined such that $(i, j) \in E$ indicates that information flows from agent i to j . Such a setup could, for example, describe a situation in which the sensing radius of each agent is much larger than the area in which their movement is confined to. However, for complexity and scalability reasons, each agent chooses to only keep track of a select few neighboring agents in the network. Let the set of agent i 's neighbors in the network be given by $N(i) = \{j \mid (j, i) \in E\}$. Since the focus of this work is mainly on high-level coordination strategies, it will be assumed that agents have single-integrator dynamics

$$\dot{x}_i = u_i, \tag{43}$$

where $u_i \in \mathbb{R}^2$ is the control signal for agent i .

Given some mission defined on the agent trajectory level which starts at $t = 0$ and ends at $t = T$, the goal is to generate an executable sequence of m decentralized controllers that tracks the given trajectory. The multi-agent system will transition

through the controller sequence using global clock-based switching, where the k th mode occurs during the time interval $[\tau_{k-1}, \tau_k)$, for $k = 1, \dots, m$. It is assumed that agents can only compute relative displacement vectors to neighbors and that they do not share a common coordinate frame, as is often the case in mobile robot or sensor network applications. Therefore, within each mode k , let each agent i compute its control as a linear combination of scaled and rotated displacement vectors to each of its neighbors:

$$u_i = - \sum_{j \in N(i)} r_{ijk} \text{Rot}(\theta_{ijk})(x_i - x_j), \quad \forall t \in [\tau_{k-1}, \tau_k), \quad (44)$$

with $r_{ijk} \in \mathbb{R}$ and $\theta_{ijk} \in [0, 2\pi)$ parameterizing the scaling and rotation, respectively, of the displacement vector between agents i and j . Here, $\text{Rot}(\cdot)$ is the counter-clockwise rotation matrix given by

$$\text{Rot}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \quad (45)$$

Notice that (44) gives the general form of a class of decentralized controllers for the system. For the sake of convenience, the scaling and rotation parameters associated with each mode k can be grouped together as

$$r_k = [\dots, r_{ijk}, \dots]^T \quad \text{and} \quad \theta_k = [\dots, \theta_{ijk}, \dots]^T. \quad (46)$$

By optimally selecting the parameters r_k and θ_k , as well as the global switching times τ_k , a decentralized controller sequence can be generated to minimize the tracking error.

To apply the results from Theorems 4.1.0.2 and 4.1.0.3 to this system, it is necessary to write down dynamics for the entire multi-agent system. First, define the $(2N \times 2N)$ adjacency matrix $A_k(r_k, \theta_k)$ associated with the k th mode, in terms of (2×2) blocks, by

$$A_{ijk}(r_k, \theta_k) = \begin{cases} r_{ijk} \text{Rot}(\theta_{ijk}) & , \text{ if } (j, i) \in E \\ 0 & , \text{ otherwise.} \end{cases} \quad (47)$$

Next, let the $(2N \times 2N)$ degree matrix $D_k(r_k, \theta_k)$ associated with the k th mode also be defined in terms of (2×2) blocks by

$$D_{ijk}(r_k, \theta_k) = \begin{cases} \sum_{z | (z,i) \in E} r_{izk} \text{Rot}(\theta_{izk}) & , \text{ if } i = j \\ 0 & , \text{ otherwise.} \end{cases} \quad (48)$$

Finally, define the weighted Laplacian matrix $L_k(r_k, \theta_k)$ associated with the k th mode as

$$L_k(r_k, \theta_k) = D_k(r_k, \theta_k) - A_k(r_k, \theta_k). \quad (49)$$

The evolution of agent states for the entire multi-agent system is then given by

$$\dot{x} = -L_k(r_k, \theta_k)x, \quad \forall t \in [\tau_{k-1}, \tau_k), \quad (50)$$

for each mode $k = 1, \dots, m$, where $x = [x_1^T, \dots, x_N^T]^T$.

Assuming that the agents have initial state $x(0) = x_0$, the task of generating a sequence of m decentralized controllers to track some target trajectory $x_d : [0, T] \rightarrow \mathbb{R}^{2N}$ can then be formulated as an optimal control problem. The objective is to choose the parameters r_k and θ_k for each mode $k = 1, \dots, m$, as well as the global switching times $\tau_1, \dots, \tau_{m-1}$, for a system with dynamics (50) so as to minimize the cost functional

$$J = \frac{1}{2} \int_0^T \|x(t) - x_d(t)\|^2 dt. \quad (51)$$

As mentioned before, extra care must be taken to ensure that the global switching times satisfy the constraints in (29).

4.3 *Simulation: Tracking a Drumline-Inspired Dance*

A MATLAB simulation was performed in which a system of $N = 21$ agents were tasked to track the drumline-inspired multi-agent dance consisting of agent trajectories shown in Figure 1. Drumline formations are traditionally designed by choreographers to be executed in a centralized manner. The position and path taken by band

members at each moment in time have been predetermined to a high level of detail. As a result, band members spend a lot of time practicing to follow these predetermined paths. However, such an approach requires each band member to memorize paths taken throughout the entire dance sequence and have global sensing capabilities to know if they're in the correct position. Optimal decentralization is used to mimic the original routine using only decentralized control laws, i.e., requiring agents to make use of only locally-available information while executing the control laws.

The target trajectory is defined on t from 0 to $T = 10.78$. A total of $m = 23$ modes were allowed for the multi-agent system to track the desired multi-agent motion with. Therefore, all 22 switching times, as well as the parameters for all 23 modes, need to be optimized. To do so, a steepest descent with Armijo step size algorithm was used. The resulting convergence of the cost J , corresponding to the tracking error, is shown in Figure 9.

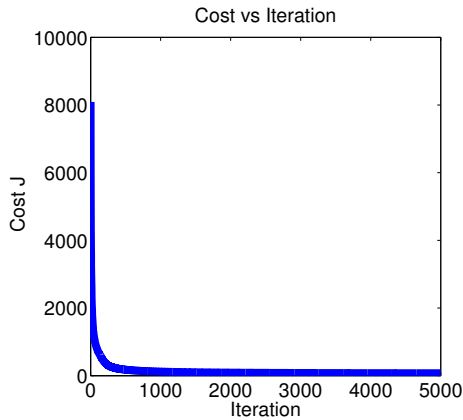


Figure 9: Convergence of cost J (tracking error) after performing steepest descent with Armijo step size on parameters and switching times for tracking a multi-agent drumline-inspired dance.

Note that everything in this example up until now is meant to be done offline and in a centralized manner. Only after the optimal decentralization algorithm has converged to a desirable cost, is the information required for execution (i.e., decentralized

control laws and switching times) downloaded to each agent. The agents then simultaneously start executing their respective control strategies using only information that is locally available in the network and switch between controllers simultaneously. The timer-based switching can occur in one of two ways. Either all agents have accurate synchronized clocks and can therefore determine when to switch controllers on their own, or a single leader agent with an accurate clock must broadcast switching signals throughout the network.

The agent trajectories resulting from a simulation where they execute the generated decentralized controller sequence for tracking the drumline-inspired dance are shown in Figure 10. Here, the actual locations of the agents are marked by O's with lines connecting them to their desired location marked by X's to help ease the comparison. Note that while the original simulation of the multi-agent dance in Figure 1 used performed using centralized coordination strategies, Figure 10 shows agents tracking the trajectories but using only decentralized controllers. Therefore, this example helps showcase the performance of the optimal decentralization algorithm in generating controllers for agents to track complex multi-agent motions.

To summarize, this chapter explored the problem of generating decentralized controllers which allowed for agents with a static network topology to best track a multi-agent motion with. By posing the problem as an optimal control problem, parameterized decentralized control laws and switching times were optimized so as to minimize a tracking error in the form of a cost functional. The resulting optimal decentralization algorithm was then showcased in a simulation where agents that could only sense relative displacement vectors between its neighbors had to work together and track a complex drumline-inspired dance trajectory.

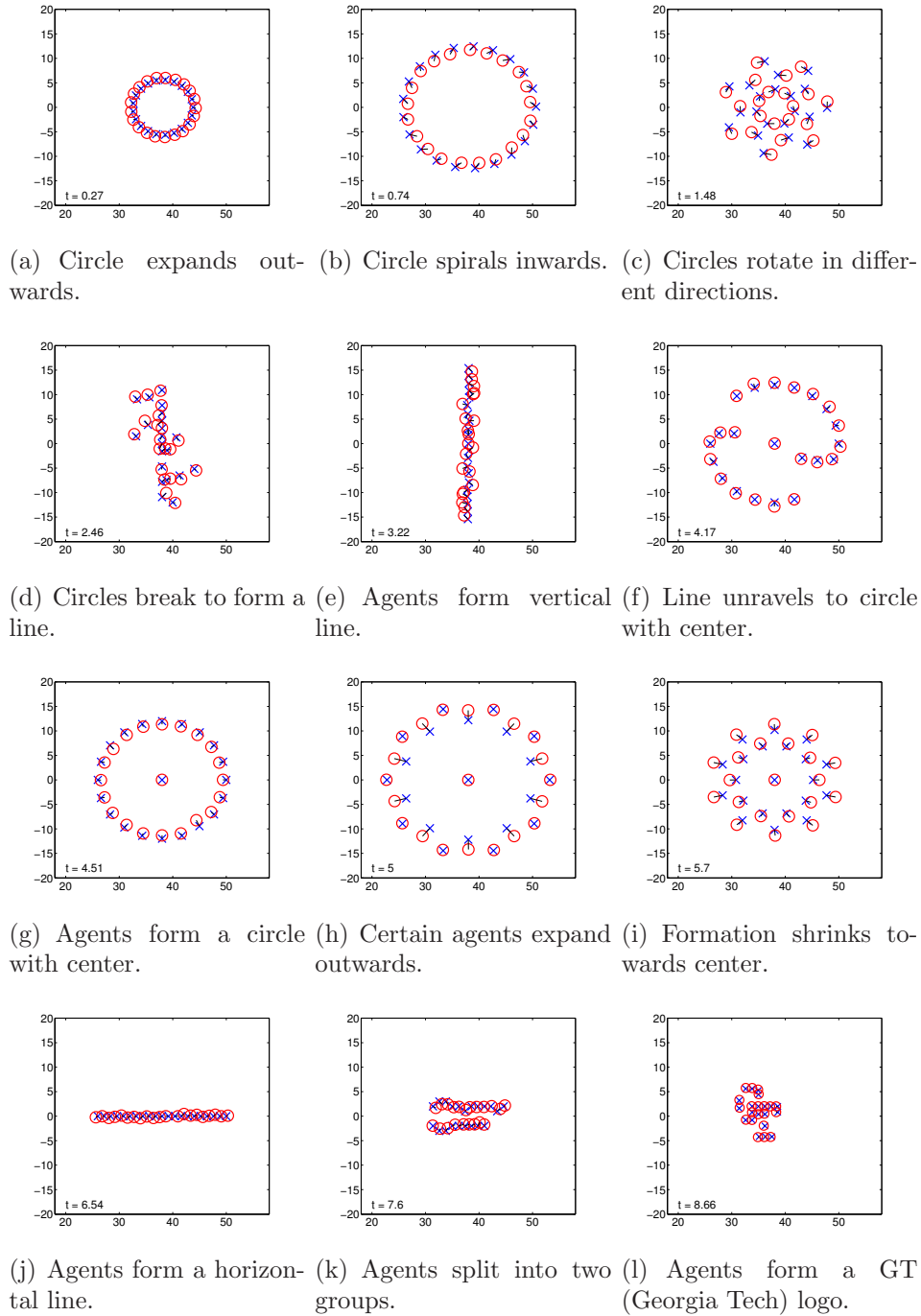


Figure 10: Simulation of $N = 21$ agents executing the drumline-inspired dance from Figure 1 using a decentralized controller sequence generated by the optimal decentralization algorithm. The locations of the agents are marked by O's with lines connecting them to their desired location marked by X's.

CHAPTER V

SCRIPTING DECENTRALIZED CONTROLLER SEQUENCES

Having addressed how to generate decentralized control laws for multi-agent systems, the next step is to see how simple control laws can be combined together to form more complex decentralized coordination strategies. As mentioned in Section 1.1, many large-scale multi-agent missions can be partitioned into a sequence of subtasks in order to simplify controller design. Take, for example, the drumline-inspired multi-agent dance from the previous chapter. One way to perform the entire mission is to have agents switch consecutively between specialized decentralized controllers for each subtask, e.g., forming a circle, expanding the circle, and spiraling in. To support such an approach to controller design, many techniques, such as MDLs (see Section 1.2.4 for more examples), exist already for doing high-level scripting of control strategies.

However, one factor which has been overlooked is that just because a decentralized controller works well in a stand-alone situation, does not necessarily mean that it will perform as expected when strung together consecutively with other controllers. This is because decentralized controllers may oftentimes have network topological requirements that must be satisfied in order for it to achieve the desired effect during execution. For example, the convergence properties associated with nearest-neighbor averaging (e.g., [83]) are based on the assumption that the network topology is a connected graph. To complicate things further, the network topology itself may be state-dependent or can change with time. Therefore, one cannot always hope to accomplish the original mission by having agents naively switch through executing the controllers for each subtask, since the network topology resulting from the termination

of one controller may not be what the next controller in the sequence needs in order to achieve its desired effects on the system during execution.

To address this problem in stringing together multiple decentralized controllers, this chapter presents the Graph Process Specification (GPS) framework from [125, 126] along with a series of detailed examples demonstrating its usage in the design of multi-agent control strategies. The GPS framework acts as a way to script sequences of decentralized controllers for agents, while simultaneously ensuring that network topological requirements are satisfied for each controller in the sequence during execution. To do so, GPS builds sequences of decentralized controllers out of fundamental building blocks called atoms. Each atom explicitly states a network topological transition (a so-called graph process as discussed in [72]). Moreover, every atom specifies the means to make this transition occur by providing a decentralized multi-agent controller, as well as a locally-checkable condition, which allows for an agent to check that the transition has taken place before terminating the controller.

Using atoms, scripting sequences of controllers in GPS reduces to selecting a sequence of atoms from a library such that each atom terminates with a network topology which the next atom in the sequence needs in order to have its desired effect on the system. Therefore, control strategy design is closely related to the mode sequencing problem seen in the control of hybrid systems, but with additional constraints on the ordering of modes. Moreover, interrupt conditions can be scripted to further specify how agents will switch through the sequence of controllers, thus making control design more related to the mode scheduling problem in hybrid systems instead but with additional constraints in place. Together, the atom sequence and interrupts form a decentralized control strategy for the agents.

Upon checking that the control strategy respects the network topological requirements for each controller during execution, it is then downloaded onto the agents. Therefore, it is possible for agents to have special a priori designations in GPS. To

carry out the control strategy, agents start by executing the controller for the first atom. This continues until an agent detects that both the required transition in the network topology has taken place, and that the interrupt condition is satisfied. Upon doing so, that agent then broadcasts a message throughout the network and all agents switch simultaneously to executing the controller for the next subtask, and so on. Thus, a multi-agent system which follows a control strategy scripted using GPS behaves as a hybrid system.

This chapter is organized as follows: First, Section 5.1 will present the technical details behind the GPS framework. Then, Section 5.2 will give examples that illustrate how GPS can be used to script executable sequences of decentralized controllers that let agents perform complex multi-agent motions. Finally, Section 5.3 connects GPS with the decentralized controller generation tool presented in Chapter 4 by showing how to use the optimal decentralization algorithm to populate a library of atoms for usage in control strategy design with GPS.

5.1 Graph Process Specification Formulation

5.1.1 Networked System Representation

Consider a collection of N agents, where the state x^i of the i th agent belongs in the differentiable manifold X , for $i \in \mathcal{N} = \{1, \dots, N\}$. Additionally, let $x \in X^N$ be the concatenated states of all N agents in the system, such that $x = [(x^1)^T \dots (x^N)^T]^T$. The network topology which describes the flow of information amongst agents at each instant will be represented by a vector-weighted directed graph $G = (\mathcal{N}, E, w)$, where $E \subseteq \mathcal{E} = \{(i, j) \mid i, j \in \mathcal{N} \text{ and } i \neq j\}$ and $(i, j) \in E$ represents a flow of information from agent i to j . Furthermore, let $w : E \rightarrow \mathbb{R}^p$, for some variable $p \in \mathbb{N}$, be a function which characterizes the information flow from one agent to another by assigning a vector to each edge. For example, in a team of mobile robots, the existence of an edge $(i, j) \in E$ means that agent j can sense agent i , while the vector-weight $w((i, j))$ may

be used to describe the associated relative displacement vector which agent j senses. We will refer to this vector-weighted directed graph as the current *information flow graph* of the network. It should be noted that the vector-weight does not necessarily have to represent the information which is sensed. For example, the vector-weight can also describe the type or strength of a flow of information between two agents in the network.

We will represent the set of all possible information flow graphs that can describe the network as $\mathcal{G} = \{(\mathcal{N}, E, w) \mid E \subseteq \mathcal{E} \text{ and } \exists p \in \mathbb{N} \text{ where } w : E \rightarrow \mathbb{R}^p\}$. Let the mapping $s : X^N \rightarrow \mathcal{G}$ be a *graph inducing function* that takes in the states of all agents, and returns the information flow graph describing the network. Furthermore, let $\mathcal{S} = \{s \mid s : X^N \rightarrow \mathcal{G}\}$ be the set of all possible graph inducing functions. Note that this formulation is similar to that of connectivity graphs in [77].

5.1.2 Atoms and Consistency

Many decentralized controllers have prerequisites on the information flow graph which must be met, when being executed by agents, in order to have the intended effect on a system. These same requirements are also what may prevent agents from naively executing an arbitrarily chosen sequence of controllers consecutively. This is because the information flow graph resulting at the termination of one controller may not necessarily be what the next controller in the sequence requires. Therefore, if one wishes to script a sequence of decentralized controllers for agents to execute, it is necessary to make explicit both the information flow graph that a controller needs, and how the information flow graph is affected as agents execute the controller.

To make this information readily available, we present the concept of *atoms*, which act as the fundamental building blocks in the GPS framework. An atom contains three key pieces of information. First, each atom describes the types of multi-agent systems that it is making a statement about, i.e., those with agent dynamics in the set \mathbb{F} and

information flow graph given by a graph inducing function from the set \mathbb{S} . Second, each atom explicitly states a transition in the system's information flow graph from the initial set \mathbb{G} to the final set \mathbb{H} . Finally, each atom describes the means by which to make this transition occur by stating a control law \mathcal{U} that agents execute, as well as a condition \mathcal{C} which lets agents detect if the transition has occurred. Formally, an atom is defined as follows:

Definition 5.1.2.1. *An atom \mathcal{A} is a tuple given by*

$$\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C}),$$

such that

1. $\mathbb{S} \subseteq \mathcal{S}$
2. $\mathbb{F} \subseteq \{f \in \mathcal{F} \mid f : X^N \times U^N \times \mathbb{R}_{\geq 0} \rightarrow (\bigcup_{x \in X} T_x X)^N\}$
3. $\mathbb{G} \subseteq \mathcal{G}$
4. $\mathbb{H} \subseteq \mathcal{G}$
5. $\mathcal{U} : \mathcal{S} \times X^N \times \mathbb{R}_{\geq 0} \rightarrow U^N$
6. $\mathcal{C} : \mathcal{S} \times X^N \times \mathbb{R}_{\geq 0} \rightarrow \{0, 1\}^N$.

Here, \mathcal{F} is the set of all functions that are Lipschitz continuous in its first two arguments and piecewise continuous in its third argument, U is a manifold corresponding to the set of control inputs, and $T_x X$ is the tangent space of a point $x \in X$.

An atom is *consistent* if the transition of the information flow graph from set \mathbb{G} to set \mathbb{H} is guaranteed to occur in finite time, and can always be detected by at least one agent in the network using the condition \mathcal{C} in finite time as well. Such a guarantee is important when designing a sequence of controllers using atoms since agents should not stop executing a controller until the information flow graph first

makes the described transition. Ensuring that the transition occurs in finite time prevents a controller in the sequence from blocking others. Moreover, requiring that at least one agent can detect the transition ensures that \mathcal{C} is effective, and lets that agent broadcast a message throughout the network for synchronous controller switching.

To formally define a consistent atom, we make use of the following shorthand notation: for a function $z : A \rightarrow B^N$, where A and B are arbitrarily defined sets, let $z^i : A \rightarrow B$, for $i \in \mathcal{N}$, be such that $\forall a \in A$, $z(a) = [(z^1(a))^T \dots (z^N(a))^T]^T$. The definition of a consistent atom is then given as follows:

Definition 5.1.2.2. *An atom $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$ is consistent when $\forall f \in \mathbb{F}$, $\forall x_0 \in X^N$, and $\forall s \in \mathbb{S}$ such that $s(x_0) \in \mathbb{G}$, if*

1. $x(t_0) = x_0$ for some $t_0 \in \mathbb{R}_{\geq 0}$
2. $\dot{x}(t) = f(x(t), \mathcal{U}(s, x(t), t), t)$,

then $\mathcal{C}^i(s, x(t), t) = 1$, for some $t \geq t_0$ and $i \in \mathcal{N}$, implies that $s(x(t)) \in \mathbb{H}$. Furthermore, $\exists t^ \in [t_0, \infty)$ and $\exists j \in \mathcal{N}$ such that $\mathcal{C}^j(s, x(t), t) = 1 \forall t \geq t^*$.*

The above definition says the following: for an atom $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$, assume that the N agents have dynamics given by $f \in \mathbb{F}$ and information flow described using a graph inducing function $s \in \mathbb{S}$. Suppose that the information flow graph at some initial time $t = t_0$ belongs to the set of initial graphs \mathbb{G} , and that the controller \mathcal{U} is used by the agents. If \mathcal{A} is consistent, then it is guaranteed that the system will evolve such that the information flow graph enters and stays in the set of final graphs \mathbb{H} within finite time. Furthermore, membership of the current information flow graph in \mathbb{H} can be locally detected, as indicated by when $\mathcal{C}^i \rightarrow 1$, for some agent $i \in \mathcal{N}$. A consistent atom requires that at least one agent in the network realize within finite time when the information flow graph has entered into and will stay in \mathbb{H} .

Note that the definition of a consistent atom allows for the control law \mathcal{U} and termination condition \mathcal{C} to be computed by an agent using information from anyone

else in the network, even if no edge exists between the two agents in the information flow graph. To respect the limitations on inter-agent information flow as described by the network topology, it is necessary to restrict each agent's computations to use only locally available information in the network. Note that such an approach follows closely with the definition of a distributed algorithm from [63]. We therefore define a function that describes an agent computation as being *decentralized* if it satisfies the following conditions:

Definition 5.1.2.3. *A function $\zeta : \mathcal{S} \times X^N \times \mathbb{R}_{\geq 0} \rightarrow B^N$ is decentralized if $\zeta^i(s, x, t) \neq \zeta^i(s, y, t)$ implies that either $x^i \neq y^i$, or there exists a $j \in \mathcal{N}$ where $(j, i) \in E$ and $x^j \neq y^j$. Here, B is some nonempty set and the edge set E comes from $s(x) = (\mathcal{N}, E, w)$. Moreover, the above must hold for all $i \in \mathcal{N}$, $s \in \mathcal{S}$, and $t \in \mathbb{R}_{\geq 0}$, and $x \in X^N$.*

In the above definition, a function ζ is decentralized if for each agent $i \in \mathcal{N}$, the evaluation of ζ^i is independent of the states of agents in the network whom are not agent i 's neighbors. Therefore, if the evaluation of ζ^i changes, then either agent i 's state or one of its neighbors' state has changed. With such a definition in place, it is now possible to enforce that the computations described by a consistent atom use only locally available information in the network. To do so, both the controller \mathcal{U} and termination condition \mathcal{C} must be decentralized.

Definition 5.1.2.4. *A decentralized consistent atom $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$ is a consistent atom where both \mathcal{U} and \mathcal{C} are decentralized.*

Before proceeding onwards, we will solidify the concepts presented thus far through an example of constructing a decentralized consistent atom for agents to perform nearest-neighbor averaging. It should be noted that a rich literature exists on nearest-neighbor averaging controllers or consensus protocols for multi-agent systems, each with their own associated merits and demerits (see Section 1.2.1). The controller

used in this example was chosen merely to illustrate the process of encapsulating a decentralized controller into a decentralized consistent atom.

Example 5.1.2.1. *Suppose that in a team of N agents, the i th agent has state $x^i \in X = \mathbb{R}^2$ that describes its planar position in Cartesian coordinates, for all $i \in \mathcal{N}$. Since we are interested in focusing on high-level coordination strategies, agents will be treated as point particles with single integrator dynamics $f_{\mathcal{I}} : \mathbb{R}^{2N} \times \mathbb{R}^{2N} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^{2N}$, where*

$$\dot{x} = f_{\mathcal{I}}(x, u, t) = u. \quad (52)$$

Furthermore, let

$$\mathbb{F}_{\mathcal{I}} = \{f_{\mathcal{I}}\} \quad (53)$$

be the set containing the agents' single integrator dynamics.

Suppose that each agent is equipped with omnidirectional sensors (e.g., sonar, LIDAR, etc.) and can measure relative displacement vectors to other agents within a radius $\delta > 0$. We will use the notion of a Δ -disk proximity graph to describe such a scenario. In particular, let $s_{\Delta}(\delta) \in \mathcal{S}$ be the graph inducing function such that $\forall x \in \mathbb{R}^{2N}$, $s_{\Delta}(\delta)(x) = (\mathcal{N}, E(x), w(x))$, with $(i, j) \in E(x)$ only when $\|x^i - x^j\| \leq \delta$. The edge weight function $w(x)$ will be used to describe the sensed distance between two neighboring agents, so that for each edge $(i, j) \in E(x)$, $w(x)((i, j)) = \|x^i - x^j\|$. For the sake of notational simplicity, the remainder of this chapter will treat all information flow graphs induced by $s_{\Delta}(\delta)$ as if they were weighted undirected graphs, since $(i, j) \in E(x) \iff (j, i) \in E(x)$ and $w((i, j)) = w((j, i))$. Let

$$\mathbb{S}_{\Delta}(\delta) = \{s_{\Delta}(\delta)\} \quad (54)$$

be the set containing $s_{\Delta}(\delta)$, the Δ -disk proximity graph inducing function. Using these choices for agent dynamics and graph inducing functions, we can now begin constructing our example decentralized consistent atom for nearest-neighbor averaging.

The goal of nearest-neighbor averaging is to have agents within the network reach consensus with one another by converging to the same state. In our case, because an agent's state is its position, nearest-neighbor averaging will drive the agents to meet at the same location. To do so, each agent calculates its control using only sensed relative displacement vectors to its neighbors. For the chosen graph inducing function and agent dynamics, [48] presents the following decentralized controller for performing nearest-neighbor averaging while maintaining network connectivity:

$$\mathcal{U}_{avg}^i(\delta)(s_{\Delta}(\delta), x, t) = - \sum_{j \in N_{\rho}(i)} \frac{2\delta - \|x^i - x^j\|}{(\delta - \|x^i - x^j\|)^2} (x^i - x^j), \quad (55)$$

for $i \in \mathcal{N}$. Here, $N_{\rho}(i) = \{j \in \mathcal{N} \text{ such that } \|x^i - x^j\| < \delta\}$ refers to the index set of agent i 's neighbors in the network which are located strictly less than δ away. The controller \mathcal{U}_{avg} is decentralized since each agent computes its control signal using only relative displacement information between itself and a subset of its neighbors in the network.

[48] states that in a network where every pair of nodes has a path that connects them (i.e., a connected graph) consisting of edges with weights less than δ , then the nearest-neighbor averaging controller will drive all agent states to the same value asymptotically. Consequently, the network topology will become and stay as K_N , the complete graph with N nodes (i.e., an edge exists between every pair of agents) in finite time. Thus, the set of initial information flow graphs for the nearest-neighbor averaging atom is given by \mathbb{G}_{avg} , where

$$\begin{aligned} \mathbb{G}_{avg}(\delta) = \{(\mathcal{N}, E, w) \mid \exists e \subseteq E \text{ where } (\mathcal{N}, e) \text{ is connected} \\ \text{and } w((i, j)) < \delta, \forall (i, j) \in e\}. \end{aligned} \quad (56)$$

To build a decentralized consistent atom, agents must be able to locally detect when the current network topology is a complete graph. The triangle inequality states, for a given choice of $0 < \lambda \leq \delta$, that if two agents j and k are both neighbors of agent i and

are within distance $\frac{\lambda}{2}$ from it, then the distance between agents j and k cannot exceed λ . Therefore, agents j and k must also be neighbors of each other as well. Using this observation, we will create the decentralized function $\mathcal{C}_{avg}(\lambda)$ for locally detecting when the network topology is a complete graph, with edge weights all less than or equal to some value λ . In particular, for a given choice of $0 < \lambda \leq \delta$ and for all $i \in \mathcal{N}$, let

$$\mathcal{C}_{avg}^i(\lambda)(s, x, t) = \begin{cases} 1 & , \text{if } N(i) = \mathcal{N} - \{i\} \text{ and } \|x^i - x^j\| \leq \frac{\lambda}{2} \forall j \in N(i) \\ 0 & , \text{otherwise,} \end{cases} \quad (57)$$

where $N(i) = \{j \in \mathcal{N} \text{ such that } \|x^i - x^j\| \leq \delta\}$ is the index set of agent i 's neighbors. Finally, we describe the set of information flow graphs that agents can locally detect using $\mathcal{C}_{avg}(\lambda)$ with the set $\mathbb{H}_{avg}(\lambda)$, where

$$\mathbb{H}_{avg}(\lambda) = \{(\mathcal{N}, E, w) \mid (\mathcal{N}, E) = K_N \text{ and } w((i, j)) \leq \lambda \forall (i, j) \in E\}. \quad (58)$$

Note that K_N above refers to the complete graph with N vertices.

Having specified all the components of the atom for nearest-neighbor averaging, we will now group them together and verify that it is indeed a decentralized consistent atom:

Lemma 5.1.2.1. *The atom for agents to perform nearest-neighbor averaging:*

$$\mathcal{A}_{avg}(\lambda, \delta) = (\mathbb{S}_{\Delta}(\delta), \mathbb{F}_{\mathcal{I}}, \mathbb{G}_{avg}(\delta), \mathbb{H}_{avg}(\lambda), \mathcal{U}_{avg}(\delta), \mathcal{C}_{avg}(\lambda)), \quad (59)$$

where $0 < \lambda \leq \delta$, is a decentralized consistent atom.

Proof. Suppose a multi-agent system has dynamics $f_{\mathcal{I}}$ and information flow graph given by the graph inducing function $s_{\Delta}(\delta)$. Having agents execute the decentralized controller $\mathcal{U}_{avg}(\delta)$, when the network has an information flow graph in set $\mathbb{G}_{avg}(\delta)$, will drive all agent states to the same value asymptotically. Consequently, the information flow graph will enter and stay in the set $\mathbb{H}_{avg}(\lambda)$ within finite time, i.e., all edge weights will not exceed λ . By construction, $\mathcal{C}_{avg}(\lambda)$ is a decentralized function that allows for a single agent in the network to detect when such a transition occurred. Therefore, $\mathcal{A}_{avg}(\lambda, \delta)$ is a decentralized consistent atom. \square

5.1.3 Modes and Composability

This previous example showed how one can create a decentralized consistent atom for a specific multi-agent maneuver. With similar methods, it is possible to construct an extensive library of decentralized consistent atoms for various multi-agent motions. This corresponds to the library of decentralized controllers illustrated in Figure 2. One can then use that library to script a sequence of decentralized controllers for agents, while ensuring that the information flow graph requirements for each controller in the sequence are satisfied during execution. In particular, a sequence of decentralized consistent atoms can be scripted such that the control law in each atom is guaranteed to terminate with an information flow graph which the controller in the next atom expects when initiating. Moreover, additional *interrupt* conditions, which we will refer to as ξ , can be specified that determine when the agents synchronously switch from one executing one controller to the next. A multi-agent system following a control strategy scripted in GPS therefore acts as a hybrid system, where the mode sequence is given by the controllers \mathcal{U} contained in each of the atoms, and the guard conditions are dependent on both \mathcal{C} and ξ .

In such a hybrid system, a mode certainly cannot switch over to the next until conditions for atom consistency are met (i.e., the information flow graph has transitioned into the set \mathbb{H} for that atom). However, the interrupt condition that is specified for executing that atom should be respected as closely as possible too. Therefore, the condition for terminating a controller should be a logical AND of both the termination condition \mathcal{C} and interrupt ξ as evaluated by an agent in the network. We will refer to both the consistent atom and interrupt condition associated with it collectively as a *mode* in GPS.

Definition 5.1.3.1. A mode is denoted by the tuple

$$\mathcal{M} = (\mathcal{A}, \xi),$$

where \mathcal{A} is a consistent atom and $\xi : \mathcal{S} \times X^N \times \mathbb{R}_{\geq 0} \rightarrow \{0, 1\}^N$. Furthermore, a mode is a decentralized mode if \mathcal{A} is a decentralized consistent atom and ξ is a decentralized function.

Observe that by keeping the consistency conditions \mathcal{C} encapsulated within the atom, while letting the interrupt mapping ξ be specified separately in the mode, we are promoting the reusability of consistent atoms. For example, the same consistent atom \mathcal{A} , that makes agents shrink a circle formation indefinitely, can be used to define different modes by simply using different interrupt mappings. One mode can be created which terminates when the circle has radius smaller than 1, while another can be created that terminates when the radius is smaller than 0.01.

It is important to note that there is nothing which says that the interrupt condition ξ cannot be blocking, e.g., if the conditions for setting off the interrupt contradict those required for atom consistency. This design choice was made to give the user the most flexibility when writing scripts with atoms. Such a choice follows the approach adopted by many mainstream computer programming languages (e.g., the ability to write infinite-loops and deadlock scenarios in Java), as well as abstraction-based motion-programming languages (e.g., blocking interrupt conditions in MDL and MDLe). Therefore, it is at the discretion of the user to avoid such blocking scenarios whenever specifying interrupts.

One of the appeals to GPS is that a sequence of decentralized controllers scripted with atoms can be easily checked to see if requirements on the information flow graph are respected for each controller during its execution. To perform such a check, we introduce the concept of mode composability. We will refer to two modes as being *composable* if no matter how the controller in the first mode terminates, the resulting information flow graph is always what the controller in the second mode expects when initiating. In particular, composability requires that each member of the first mode's set of final information flow graphs \mathbb{H} belong to the second mode's set of

initial information flow graphs \mathbb{G} .

Definition 5.1.3.2. *The mode $\mathcal{M}_1 = (\mathcal{A}_1, \xi_1)$ is composable with the mode $\mathcal{M}_2 = (\mathcal{A}_2, \xi_2)$, where $\mathcal{A}_1 = (\mathbb{S}_1, \mathbb{F}_1, \mathbb{G}_1, \mathbb{H}_1, \mathcal{U}_1, \mathcal{C}_1)$ and $\mathcal{A}_2 = (\mathbb{S}_2, \mathbb{F}_2, \mathbb{G}_2, \mathbb{H}_2, \mathcal{U}_2, \mathcal{C}_2)$, if $\mathbb{H}_1 \subseteq \mathbb{G}_2$. We will denote this property by $\mathcal{M}_1 \prec \mathcal{M}_2$.*

Note that mode composability does not necessarily commute. For example, a mode that drives agents from a line formation to a circle formation may compose with a mode that rotates the circle formation, but certainly not the other way around.

5.1.4 Graph Process Specifications

To script a control strategy for agents using the GPS framework, two key pieces of information are needed. The first is the mode sequence which describes the controllers that agents will switch through executing consecutively, as well as how that switching will occur. The second is a precise description of the multi-agent system used to check if the mode sequence can indeed be executed successfully. In this description, x_0 gives the initial state information, s^* is the graph inducing function that gives the information flow graph of the system, and f^* describes the agent dynamics. These two pieces of information will be grouped together into what we call a *Graph Process Specification (GPS)*.

Definition 5.1.4.1. *A Graph Process Specification (GPS) is a tuple given by*

$$GPS = ((x_0, s^*, f^*), (\mathcal{M}_1, \dots, \mathcal{M}_m)),$$

where $m \in \mathbb{N}$ and $\mathcal{M}_k = (\mathcal{A}_k, \xi_k)$, for $k = 1, \dots, m$, are modes such that

1. $x_0 \in X^N$
2. $s^* \in \mathcal{S}$
3. $f^* \in \{f \in \mathcal{F} \mid f : X^N \times U^N \times \mathbb{R}_{\geq 0} \rightarrow (\bigcup_{x \in X} T_x X)^N\}$.

Three checks are required to verify that a sequence of controllers scripted using GPS is executable by the multi-agent system which the script was written for. First, it is necessary to first check if the atoms contained within each mode are valid for the multi-agent system of interest. To do this, one must verify that the multi-agent system's graph inducing function s^* and agent dynamics f^* fall into the sets \mathbb{S} and \mathbb{F} , respectively, for each mode's atom. Next, the initial condition of the agents have to be such that the induced information flow graph of the system allows for agents to start executing the first mode's controller. Therefore, it is necessary to check that the information flow graph $s^*(x_0)$ belongs to the set \mathbb{G} of the first mode's atom. Finally, after verifying that the first mode can be initiated, we must ensure that each mode can transition to the next while respecting the each mode's requirements on the information flow graph. Therefore, a final check must be performed to verify that each mode composes with the next in the sequence. Moreover, if each mode in the GPS is decentralized, then the multi-agent system can execute the entire sequence of controllers using only locally available information in the network, with the exception of global broadcasts for simultaneous mode switches. These requirements are described formally below:

Definition 5.1.4.2. *A GPS $((x_0, s^*, f^*), (\mathcal{M}_1, \dots, \mathcal{M}_m))$ is executable if*

1. $\mathcal{M}_k \prec \mathcal{M}_{k+1}$, for $k = 1, \dots, m - 1$
2. $s^* \in \mathbb{S}_k$, for $k = 1, \dots, m$
3. $f^* \in \mathbb{F}_k$, for $k = 1, \dots, m$
4. $s^*(x_0) \in \mathbb{G}_1$,

where $\mathcal{A}_k = (\mathbb{S}_k, \mathbb{F}_k, \mathbb{G}_k, \mathbb{H}_k, \mathcal{U}_k, \mathcal{C}_k)$, for $k = 1, \dots, m$. Furthermore, an executable GPS is locally executable if each mode \mathcal{M}_k , for $k = 1, \dots, m$, is a decentralized mode.

5.1.5 Executing Graph Process Specifications

To illustrate how agents will behave when following a sequence of controllers scripted as an executable GPS, we will formally describe its execution. We start by defining a variant of the *hybrid time sets* used in [49] to describe the time intervals in which each mode of the GPS is being executed:

Definition 5.1.5.1. *A hybrid time set is a sequence of intervals $Q = \{q_1, \dots, q_w\}$, for some $w \in \mathbb{N}$, such that*

1. $q_k = [z_k, z'_k]$, for $k = 1, \dots, w - 1$
2. $q_w = [z_w, z'_w]$ if $z'_w < \infty$, and $[z_w, \infty)$ otherwise
3. $z_k \leq z'_k$, for $k = 1, \dots, w$
4. $z'_k = z_{k+1}$ for $k = 1, \dots, w - 1$.

Hybrid time sets are used to describe the execution of a GPS similar to how [49] uses them to describe the execution of a hybrid system: as a set of requirements on the state trajectory. Therefore, a state trajectory is either accepted or rejected as an execution of the GPS. To be an execution of a GPS, the state trajectory must begin at the initial condition specified in the GPS. The state evolution in each mode must be driven by the controller in that mode's consistent atom, as applied to the agent dynamics. Lastly, each mode terminates as soon as any agent detects that the information flow graph has entered into the set of final graphs and that the interrupt conditions are satisfied as well. Although the end of a mode is detected by a single agent, all agents switch modes simultaneously.

Definition 5.1.5.2. *Given an executable GPS, $((x_0, s^*, f^*), (\mathcal{M}_1, \dots, \mathcal{M}_m))$, its execution is a pair (Q, x) , where $Q = \{q_1, \dots, q_{\tilde{m}}\}$ is a hybrid time set with $\tilde{m} \leq m$ and $z_1 = 0$. If $\tilde{m} < m$, then $z'_{\tilde{m}} = \infty$, while if $\tilde{m} = m$, then we allow for $z'_{\tilde{m}} \leq \infty$.*

Additionally, $x(t)$ is a state trajectory defined on either $t \in [0, \infty)$ if $z'_{\tilde{m}} = \infty$, or on $t \in [0, z'_{\tilde{m}}]$ if $z'_{\tilde{m}} < \infty$, such that

1. $x(0) = x_0$
2. $\dot{x}(t) = f^*(x(t), \mathcal{U}_k(s^*, x(t), t), t)$ when $t \in q_k$, for $k = 1, \dots, \tilde{m}$.
3. For each $k = 1, \dots, \tilde{m} - 1$, $\exists i \in \mathcal{N}$ such that

$$C_k^i(s^*, x(z'_k), z'_k) = 1 \text{ and } \xi_k^i(s^*, x(z'_k), z'_k) = 1.$$

If $z'_{\tilde{m}} < \infty$, then the above also holds for $k = \tilde{m}$.

4. For each $k = 1, \dots, \tilde{m}$, $\nexists t \in q_k - \{z'_k\}$ such that

$$C_k^i(s^*, x(t), t) = 1 \text{ and } \xi_k^i(s^*, x(t), t) = 1$$

for some $i \in \mathcal{N}$.

This definition describes an execution of a GPS in the following way: the state trajectory $x(t)$ of the agents starts at the initial condition x_0 at time $t = 0$. Given that the GPS contains a sequence of m modes, q_k corresponds to the time that mode k is being executed, for $k = 1, \dots, \tilde{m}$, where $\tilde{m} \leq m$. In the k th mode, as indicated by when $t \in q_k$, the agent state dynamics f^* uses the controller \mathcal{U}_k , as supplied by \mathcal{A}_k . The k th mode stops and switches to the $k + 1$ th mode in the sequence (or stops the execution of the GPS if $k = m$) the instant $t = z'_k$. This corresponds to the first time $t \in q_k$ when both $C_k^i \rightarrow 1$ and $\xi_k^i \rightarrow 1$, for any agent $i \in \mathcal{N}$. Finally, since the end of the k th mode depends on a user defined interrupt mapping ξ_k , it is possible that the interrupt never fires, causing the k th mode to continue executing forever. Such a blocking scenario is why we allow for $\tilde{m} \leq m$. Figure 11 provides an illustration showing how the execution of an executable GPS with three modes is viewed as a hybrid system.

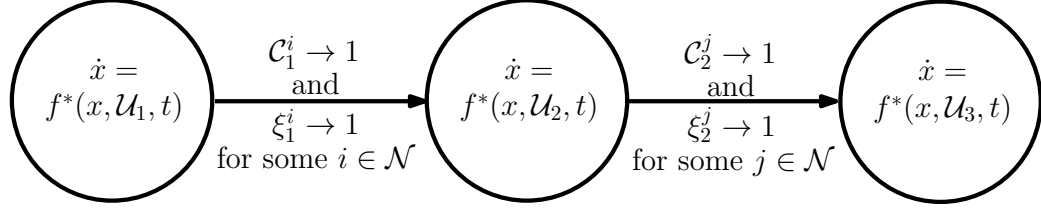


Figure 11: An illustration showing how the execution of an executable GPS with three modes can be viewed as a hybrid system.

In this section, we have defined the tools which the GPS framework is composed of, as well as the execution of an executable GPS. In the next section, we will give a detailed example of GPS can be used to script a sequence of decentralized controllers that makes agents switch between multiple formations.

5.2 Graph Process Specification Examples

In this section, we provide an example of how GPS can be used to script a sequence of decentralized controllers that makes agents first go into a line formation, and then switch into a circle formation. The process of encapsulating the relevant multi-agent controllers into decentralized consistent atoms will be illustrated in detail. Two scripts written using GPS will be created using those atoms: the original which is not executable, and a revised one which is made executable through a mode insertion. Note that this section builds off of the nearest-neighbor averaging example in Example 5.1.2.1. Therefore, all assumptions and definitions made previously will still hold true in this section.

5.2.1 Connectedness-Preserving Formation Control Laws

We will use the connectedness-preserving formation control law from [48] throughout this example as a way to have agents move into desired formations. It should be noted that many controllers exist in literature for formation control in multi-agent systems, each with their own respective merits and demerits. We use the formation control law from [48] here simply as an example to illustrate how one can encapsulate

an existing decentralized multi-agent controller into a decentralized consistent atom.

Details of the controller will now be reviewed in the context of GPS. Let the set of target points $\tau^i \in \mathbb{R}^2$, for all $i \in \mathcal{N}$, describe the desired relative displacements between agents in a formation. The formation control law uses only locally available information to drive the agents in such a way that $\|(x^i - x^j) - (\tau^i - \tau^j)\| \rightarrow 0$, for all pairs of agents $i, j \in \mathcal{N}$. In other words, the control law makes an agent compare its actual displacements with its neighbors to the displacements required to create the formation. It then makes the agents move so as to make those two displacements equal one another, thereby driving agents into a translation of the formation specified by the target points. Therefore, the need for a global coordinate system is avoided.

Have $\tau = [(\tau^1)^T \dots (\tau^N)^T]^T$ be the vector of concatenated target points. The shorthand notation $d_{ij} = \tau^i - \tau^j$, for all $i, j \in \mathcal{N}$, will be used to represent the displacement vector between any two agents i and j in the desired formation. Furthermore, let $l_{ij}(t) = x^i(t) - x^j(t)$, for all $i, j \in \mathcal{N}$, be the actual displacement vector between agents i and j at time t . A controller that lets agents achieve formations while maintaining network connectivity will now be presented.

Theorem 5.2.1.1. *Suppose a multi-agent system has graph inducing function $s_\Delta(\delta)$ and dynamics $f_{\mathcal{I}}$. Let the information flow graph induced by the target points be given by $s_\Delta(\delta)(\tau) = (\mathcal{N}, E_d, w_d)$, where (\mathcal{N}, E_d) is connected and $w_d((i, j)) < \delta$ for all $(i, j) \in E_d$. Furthermore, have the graph induced by the agent states at time t be $s_\Delta(\delta)(x(t)) = (\mathcal{N}, E(x(t)), w(x(t)))$. [48] states that if at some initial time $t = t_0$:*

1. $E_d \subseteq E(x(t_0))$,
2. $\|l_{ij}(t_0)\| \leq \epsilon^*$ for some specific $\epsilon^* > 0$ (see [48] for details), for all $(i, j) \in E_d$,

then having agents execute the control law $\mathcal{U}_{form}(E_d, \tau, \delta)$, such that

$$\begin{aligned} & \mathcal{U}_{form}^i(E_d, \tau, \delta)(s_\Delta(\delta), x, t) \\ &= - \sum_{j \in N_{G_d}(i)} \frac{2(\delta - \|d_{ij}\|) - \|l_{ij}(t) - d_{ij}\|}{(\delta - \|d_{ij}\| - \|l_{ij}(t) - d_{ij}\|)^2} (d_{ij} - l_{ij}(t)) \end{aligned} \quad (60)$$

for all $i \in \mathcal{N}$, where $N_{G_d}(i) = \{j \mid (j, i) \in E_d\}$, will guarantee that $E_d \subseteq E(x(t))$ for all $t \geq t_0$. Furthermore, the agent states $x(t)$ will converge asymptotically to the translationally-invariant formation defined by the target points τ in the sense that $\|l_{ij}(t) - d_{ij}\| \rightarrow 0$, for all $i, j \in \mathcal{N}$, as $t \rightarrow \infty$.

To reiterate, if agents are initially close enough to one another and the initial network topology is a supergraph of the graph induced by the target points, then it will remain a supergraph while the controller $\mathcal{U}_{form}(E_d, \tau, \delta)$ is executed. Furthermore, the controller will make agents move asymptotically into the desired formation using only locally available information within the network. Therefore, $\mathcal{U}_{form}(E_d, \tau, \delta)$ is a decentralized controller. It should be noted that this controller drives agents to a desired formation assuming that each agent has an a priori assignment (i.e., agent i knows it should go to position i in the formation). Moreover, it requires agents to store information about the target points τ so that each agent knows its required relative displacement to other agents in the network. Next, we will specialize this controller to make agents go into line and circle formations, as well as encapsulate them within decentralized consistent atoms to be used in the GPS framework.

5.2.2 Line-Formation Decentralized Consistent Atom

We start by constructing a decentralized consistent atom that drives N agents into a line formation. Let τ_{line} , the vector of concatenated target points describing the desired formation, be given by

$$\tau_{line}^1 = [0 \ 0]^T \text{ and } \tau_{line}^{i+1} = \tau_{line}^i - [0.9\delta \ 0]^T, \quad (61)$$

for $i = 1, \dots, N-1$. The information flow graph resulting from applying $s_\Delta(\delta)$ to τ_{line} is the line graph $G_{line} = (\mathcal{N}, E_{line}, w_{line})$ where $E_{line} = \{(i, i+1) \mid i = 1, \dots, N-1\}$ and $w_{line}((i, j)) = 0.9\delta$, for all $(i, j) \in E_{line}$. To better visualize the desired line formation, the location of the target points, along with the corresponding network topology, are illustrated in Figure 12 for $N = 6$ and $\delta = 1$.

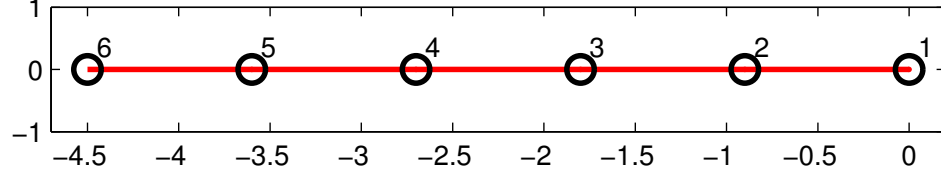


Figure 12: The target points τ_{line} and the network topology (\mathcal{N}, E_{line}) for the line formation, with $N = 6$ and $\delta = 1$.

To have agents successfully use the formation control law in Theorem 5.2.1.1, we require that the information flow graph induced by the agent states initially belong to

$$\begin{aligned} \mathbb{G}_{line}(\epsilon_{line}^*) &= \{(\mathcal{N}, E, w) \mid E_{line} \subseteq E \text{ and} \\ &\quad w((i, j)) \leq \epsilon_{line}^* \text{ for all } (i, j) \in E_{line}\}, \end{aligned} \quad (62)$$

where ϵ_{line}^* is chosen appropriately. The desired line formation can then be achieved by specializing the formation control law (60) to create a new control law $\mathcal{U}_{line}(\delta)$, where

$$\mathcal{U}_{line}^i(\delta)(s, x, t) = \mathcal{U}_{form}^i(E_{line}, \tau_{line}, \delta)(s, x, t), \quad (63)$$

for all $i \in \mathcal{N}$.

Although ideally we would like for the agents to stop executing the controller when they have perfectly achieved the line formation, the controller cannot guarantee that it occurs in finite time. Furthermore, checking to see whether the network topology has become a line graph is difficult for a single agent to do in a decentralized manner. For the sake of this example, we instead let the set of final graphs contain information flow graphs that can be easily checked by a single agent, i.e., when agent 1's only neighbor is agent 2. A timer interrupt can then be used later on, when constructing modes from this atom, to delay the controller's termination and allow the agents to get arbitrarily close to the desired formation. Thus, the set of final information flow

graphs for the controller is chosen to be

$$\begin{aligned} \mathbb{H}_{line}(\delta) = & \{(\mathcal{N}, E, w) \mid E_{line} \subseteq E, w((i, j)) < \delta \text{ for all } (i, j) \in E_{line}, \\ & \text{and } (1, j) \notin E, \forall j \neq 2\}, \end{aligned} \quad (64)$$

In the set of final graphs above, the requirements on the edge weights are guaranteed to be met by the connectedness-preserving nature of the controller. For simplicity, we will let agent 1 be the only one that can detect the transition of the information flow graph into the set $\mathbb{H}_{line}(\delta)$. To do so, agent 1 will be using the decentralized function \mathcal{C}_{line} , where

$$\mathcal{C}_{line}^1(s, x, t) = \begin{cases} 1 & , \text{ if } N(1) = \{2\} \\ 0 & , \text{ otherwise,} \end{cases} \quad (65)$$

with $N(1)$ being the index set of agent 1's neighbors in the induced graph. Furthermore, since only agent 1 will be checking for the termination of the controller in this example, we let

$$\mathcal{C}_{line}^i(s, x, t) = 0, \text{ for } i = 2, \dots, N. \quad (66)$$

With all the components in place, we are ready to construct the decentralized consistent atom for driving agents into a line formation.

Lemma 5.2.2.1. *The atom for driving agents into a line formation:*

$$\mathcal{A}_{line}(\epsilon_{line}^*, \delta) = (\mathbb{S}_{\Delta}(\delta), \mathbb{F}_{\mathcal{I}}, \mathbb{G}_{line}(\epsilon_{line}^*), \mathbb{H}_{line}(\delta), \mathcal{U}_{line}(\delta), \mathcal{C}_{line}), \quad (67)$$

where ϵ_{line}^* is chosen to satisfy Theorem 5.2.1.1, is a decentralized consistent atom.

Proof. To check the consistency of the atom, we assume the information flow graph is initially in $\mathbb{G}_{line}(\epsilon_{line}^*)$. Since $\mathcal{U}_{line}(\delta)$ is the formation control law from Theorem 5.2.1.1, it will asymptotically drive the agents to the formation specified by τ_{line} . Upon getting close enough to the desired formation, the information flow graph $s_{\Delta}(\delta)(x(t))$ becomes and stays as G_{line} , and therefore transitions into the set $\mathbb{H}_{line}(\delta)$ in finite time. By construction, \mathcal{C}_{line} allows for agent 1 to locally check if the transition has occurred. \square

5.2.3 Circle-Formation Decentralized Consistent Atom

Next, we will construct a decentralized consistent atom that makes N agents go into a circle formation using a design similar to that used for $\mathcal{A}_{line}(\epsilon_{line}^*, \delta)$. Let the target points $\tau_{circ}^i \in \mathbb{R}^2$, for $i = 1, \dots, N$, which describe the desired formation, be given by

$$\tau_{circ}^1 = [0 \ 0]^T \quad (68)$$

and

$$\tau_{circ}^{i+1} = \tau_{circ}^i + Rot\left(\frac{2\pi}{N}\right)^i [0.9\delta \ 0]^T, \quad (69)$$

for $i = 1, \dots, N - 1$, where $Rot(\cdot)$ designates the counterclockwise rotation matrix

$$Rot(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \quad (70)$$

Furthermore, have $\tau_{circ} = [(\tau_{circ}^1)^T \dots (\tau_{circ}^N)^T]^T$ be the concatenated target points.

The information flow graph resulting from applying $s_\Delta(\delta)$ to τ_{circ} is the cycle graph $G_{circ} = (\mathcal{N}, E_{circ}, w_{circ})$ where $E_{circ} = \{(N, 1)\} \cup \{(i, i + 1) \mid i = 1, \dots, N - 1\}$ and $w_{circ}((i, j)) = 0.9\delta$, for all $(i, j) \in E_{circ}$. To better visualize the desired circle formation, Figure 13 shows the locations of the target points in τ_{circ} , along with the corresponding network topology, for $N = 6$ and $\delta = 1$.

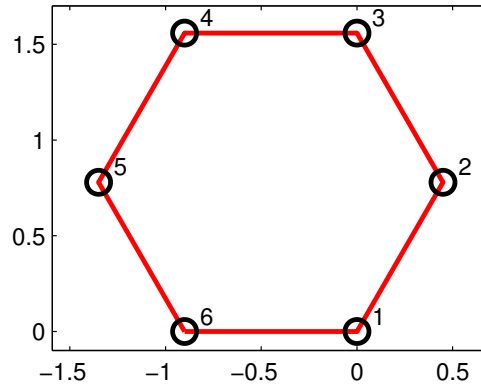


Figure 13: The target points τ_{circle} and the network topology $(\mathcal{N}, E_{circle})$ for the circle formation, with $N = 6$ and $\delta = 1$.

Just like with the line formation case, we assume that the information flow graph is initially in

$$\begin{aligned} \mathbb{G}_{circ}(\epsilon_{circ}^*) &= \{(\mathcal{N}, E, w) \mid E_{circ} \subseteq E \text{ and} \\ &w((i, j)) \leq \epsilon_{circ}^* \text{ for all } (i, j) \in E_{circ}\}. \end{aligned} \quad (71)$$

We specialize the control law (60) to achieve the desired circle formation by defining the controller $\mathcal{U}_{circ}(\delta)$, where

$$\mathcal{U}_{circ}^i(\delta)(s, x, t) = \mathcal{U}_{form}^i(E_{circ}, \tau_{circ}, \delta)(s, x, t). \quad (72)$$

For the same reasons as when designing $\mathcal{A}_{line}(\epsilon_{line}^*, \delta)$, we only require that the set of final graphs be:

$$\begin{aligned} \mathbb{H}_{circ}(\delta) &= \{(\mathcal{N}, E, w) \mid E_{circ} \subseteq E, w((i, j)) < \delta \text{ for all } (i, j) \in E_{circ}, \\ &\text{and } N(1) = \{2, 6\}\}, \end{aligned} \quad (73)$$

and use the decentralized function \mathcal{C}_{circ} to have agent 1 detect when the induced graph has entered $\mathbb{H}_{circ}(\delta)$ using locally available information, where

$$\mathcal{C}_{circ}^1(s, x, t) = \begin{cases} 1 & , \text{ if } N(1) = \{2, 6\} \\ 0 & , \text{ otherwise,} \end{cases} \quad (74)$$

and

$$\mathcal{C}_{circ}^i(s, x, t) = 0, \text{ for } i = 2, \dots, N. \quad (75)$$

Combining all the components defined thus far creates the decentralized consistent atom that drives agents into a circle formation.

Lemma 5.2.3.1. *The atom for driving agents to a circle formation:*

$$\mathcal{A}_{circ}(\epsilon_{circ}^*, \delta) = (\mathbb{S}_\Delta(\delta), \mathbb{F}_I, \mathbb{G}_{circ}(\epsilon_{circ}^*), \mathbb{H}_{circ}(\delta), \mathcal{U}_{circ}(\delta), \mathcal{C}_{circ}), \quad (76)$$

where ϵ_{circ}^* is chosen to satisfy Theorem 5.2.1.1, is a decentralized consistent atom.

Proof. The proof is identical to that for $\mathcal{A}_{line}(\epsilon_{line}^*, \delta)$ in Lemma 5.2.2.1. \square

5.2.4 Locally Executable GPS Example

So far in the chapter, a total of three decentralized consistent atoms have been constructed: nearest-neighbor averaging (Lemma 5.1.2.1), line formation (Lemma 5.2.2.1), and circle formation (Lemma 5.2.3.1). Together, these three atoms make a simple atom library which we can use in this example for scripting sequences of decentralized controllers for agents. Note that the atom library here corresponds to the library of decentralized controllers as shown in the design methodology flowchart in Figure 2. Consider now, the task of designing a sequence of controllers that will make agents first go into a line formation, and then switch into a circle formation. We will show how to use the atoms already existing in our simple atom library to achieve this. It should be noted that for illustrative purposes, the sequence of atoms which agents will execute to do this task will be scripted manually. It is recommended that for more complex tasks and where the atom library is larger, that existing techniques discussed in Section 1.2.4 on mode sequencing and scheduling for hybrid systems be used instead.

To begin, we start by examining the specific multi-agent system that the script will be written for. In particular, the multi-agent system will consist of $N = 6$ agents, with single-integrator dynamics $f_{\mathcal{I}}$, and initial positions $x_0 \in \mathbb{R}^{12}$, given by $x_0 = [(x_0^1)^T \dots (x_0^6)^T]$, where

$$\begin{aligned} x_0^1 &= [0 \ 0]^T, & x_0^2 &= [0.25 \ 0.25]^T, & x_0^3 &= [0.25 \ 0.55]^T, \\ x_0^4 &= [0 \ 0.37]^T, & x_0^5 &= [-0.37 \ 0.37]^T, & x_0^6 &= [-0.25 \ 0.75]^T. \end{aligned}$$

Suppose that each agent can sense neighboring agents which are located within a radius of $\delta = 1$. The information flow graph of the network is therefore described by the function $s_{\Delta}(1)$, and the induced graph of the agents' initial states forms a complete graph where all edge weights are less than 0.8.

Now that we know the system which we will be designing a sequence of controllers for, the next step is to script a sequence of atoms to be executed. Note that

the overall mission for the agents can be broken down into two subtasks: form a line, and then form a circle. Such a partitioning of the mission corresponds to the subtask planning stage of the design methodology in Figure 2. Our first attempt at constructing an atom sequence is to use the naive approach of using one atom to accomplish each of the subtasks, and simply executing the two back to back. Letting ϵ_{line}^* and ϵ_{circ}^* from Lemmas 5.2.2.1 and 5.2.3.1 equal 0.8, we get that $s_{\Delta}(1)(x_0) \in \mathbb{G}_{line}(0.8)$ and $s_{\Delta}(1)(x_0) \in \mathbb{G}_{circ}(0.8)$. The first proposed atom sequence is therefore given by $\mathcal{A}_{line}(0.8, 1)$, followed immediately by $\mathcal{A}_{circ}(0.8, 1)$.

To place the two atoms into modes, it is necessary to define interrupts ξ_{line} and ξ_{circ} for the line and circle atoms respectively. Recall that the final graph sets $\mathbb{H}_{line}(\delta)$ and $\mathbb{H}_{circ}(\delta)$ allow for their associated controllers to terminate execution before the agents have perfectly formed the desired formation. Such a design choice was made because of the asymptotic nature of the control laws, in which the agents will only continuously get closer to the desired formation but never perfectly achieve it. To provide the agents with an adequate amount of time to form each formation in a visually appealing way, we will define the interrupts such that $\xi_{line}^i \rightarrow 1$ and $\xi_{circ}^i \rightarrow 1$, for all $i \in \mathcal{N}$ and for all time, after 3 seconds have elapsed since they were first evaluated. Combining these interrupts with the decentralized consistent atoms yields the following decentralized modes:

$$\mathcal{M}_{line} = (\mathcal{A}_{line}(0.8, 1), \xi_{line}) \text{ and } \mathcal{M}_{circ} = (\mathcal{A}_{circ}(0.8, 1), \xi_{circ}). \quad (77)$$

Combining the mode sequence with a description of the multi-agent system that is expected to execute the controllers yields GPS_1 , where

$$GPS_1 = ((x_0, s_{\Delta}(1), f_{\mathcal{I}}), (\mathcal{M}_{line}, \mathcal{M}_{circ})) \quad (78)$$

Checking GPS_1 reveals that it is *not executable* because the mode \mathcal{M}_{line} does not compose with \mathcal{M}_{circ} . This is because any graph in $\mathbb{H}_{line}(1)$ has agent 2 being the only neighbor of agent 1, whereas all graphs in $\mathbb{G}_{circ}(0.8)$ require that agents 1 and 6 be

neighbors as well. Furthermore, some graphs belonging to $\mathbb{H}_{line}(1)$ have edge weights that are too large to belong in $\mathbb{G}_{circ}(0.8)$.

To fix these problems, we will consider inserting a mode between \mathcal{M}_{line} and \mathcal{M}_{circ} that adds additional edges to the information flow graph and decreases all of the edge weights. Fortunately, the decentralized consistent atom for nearest-neighbor averaging from Example 5.1.2.1 does exactly what is needed in this situation. Using $\mathcal{A}_{avg}(\lambda, \delta)$, we can define the decentralized mode

$$\mathcal{M}_{avg} = (\mathcal{A}_{avg}(0.8, 1), \xi_{avg}), \quad (79)$$

where $\xi_{avg}^i \rightarrow 1$ always, for all $i \in \mathcal{N}$. A new script can then be written, where the mode \mathcal{M}_{avg} is inserted between the two existing modes \mathcal{M}_{line} and \mathcal{M}_{circ} . Therefore, the script makes agents first go into a line formation, then perform nearest-neighbor averaging, and finally go into a circle formation. The new mode sequence, along with a description of the multi-agent system, are combined to form GPS_2 which is shown below to be locally executable.

Lemma 5.2.4.1. *The graph process specification*

$$GPS_2 = ((x_0, s_\Delta(1), f_I), (\mathcal{M}_{line}, \mathcal{M}_{avg}, \mathcal{M}_{circ})) \quad (80)$$

is locally executable.

Proof. First, note that $s_\Delta(1)$ and f_I belong to the respective graph inducing function and agent dynamics sets in the decentralized consistent atoms of all three modes. Since $s_\Delta(1)(x_0)$ gives a complete graph where all edge weights are less than 0.8, $s_\Delta(1)(x_0) \in \mathbb{G}_{line}(0.8)$. Noticing that each graph in $\mathbb{H}_{line}(1)$ has a line graph as a subgraph, with edge weights strictly less than 1, we see that $\mathbb{H}_{line}(1) \subseteq \mathbb{G}_{avg}(1)$. Lastly, because $\mathbb{H}_{avg}(0.8)$ only contains complete graphs with edge weights less than or equal to 0.8, each graph contains the required cycle subgraph to belong in $\mathbb{G}_{circ}(0.8)$

and so $\mathbb{H}_{avg}(0.8) \subseteq \mathbb{G}_{circ}(0.8)$. These checks show that GPS_2 is executable. Furthermore, since each of the modes in GPS_2 are decentralized modes, it is also locally executable. \square

A simulation of agents executing the sequence of decentralized controllers scripted by the locally executable GPS_2 is shown in Figure 14, for $N = 6$ and $\delta = 1$.

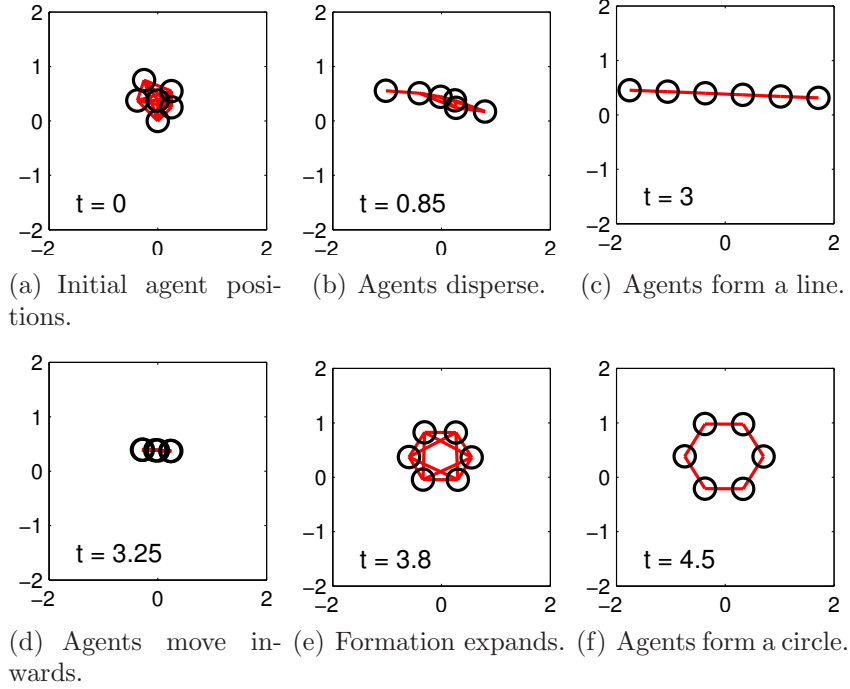


Figure 14: Simulation of agents executing the sequence of decentralized controllers scripted by the locally executable GPS_2 , as given in (80), for $N = 6$ and $\delta = 1$. The location of the agents are marked by O's and the lines indicate edges in the induced graph.

The strategy of inserting a mode containing the atom $\mathcal{A}_{avg}(\lambda, \delta)$ into GPS_1 's mode sequence to form the executable GPS_2 turns out to be a useful strategy which applies to many scenarios. Therefore, $\mathcal{A}_{avg}(\lambda, \delta)$ can be thought of as a “universal glue” for certain pairs of modes which are not composable. In particular, the final set of graphs in the first mode must be a subset of $\mathbb{G}_{avg}(\delta)$. Such a requirement is very reasonable since all that is required is that the first mode terminates with an information flow

graph that remains connected even in the presence of small perturbations to inter-agent displacements. Since $\mathcal{A}_{avg}(\lambda, \delta)$ adds edges to the information flow graph and decreases the edge weights, the set of initial graphs in the second mode must include all graphs after a certain number of edges have been added and all edge weights have fallen below some threshold. Such a property is described in the definition below.

Definition 5.2.4.1. *Suppose there exists a nonempty set of information flow graphs $\mathbb{G} \subseteq \mathcal{G}$, whose members all have vector-weight functions w that return scalar values. \mathbb{G} is inclusive if $(\mathcal{N}, E, w) \in \mathbb{G}$ implies that $(\mathcal{N}, \hat{E}, \hat{w}) \in \mathbb{G}$ as well, where $E \subseteq \hat{E}$ and $\hat{w}((i, j)) \leq w((i, j))$ for all $(i, j) \in E$.*

Using the definition of inclusive sets of information graphs, we can then precisely specify types of modes which are not composable but can be fixed by inserting in a mode containing the atom $\mathcal{A}_{avg}(\lambda, \delta)$.

Theorem 5.2.4.1. *Let the two modes \mathcal{M}_1 and \mathcal{M}_2 , where $\mathcal{M}_i = (\mathcal{A}_i, \xi_i)$ and $\mathcal{A}_i = (\mathbb{S}_i, \mathbb{F}_i, \mathbb{G}_i, \mathbb{H}_i, \mathbb{U}_i, \mathbb{C}_i)$, for $i = 1, 2$, be such that $\mathbb{H}_1 \subseteq \mathbb{G}_{avg}(\delta)$ and \mathbb{G}_2 is inclusive. Then there exists a $\lambda^* > 0$ where if $\mathcal{M} = (\mathcal{A}_{avg}(\lambda^*, \delta), \xi)$, and ξ is any arbitrary interrupt mapping, then $\mathcal{M}_1 \prec \mathcal{M}$ and $\mathcal{M} \prec \mathcal{M}_2$.*

Proof. $\mathcal{M}_1 \prec \mathcal{M}$ follows from the assumption that $\mathbb{H}_1 \subseteq \mathbb{G}_{avg}(\delta)$. Since \mathbb{G}_2 is inclusive, it must contain all information flow graphs that are complete and have edge weights less than or equal to some threshold λ^* . Therefore, $\mathbb{H}_{avg}(\lambda^*) \subseteq \mathbb{G}_2$ and so $\mathcal{M} \prec \mathcal{M}_2$. □

To summarize, we started out with a mission consisting of two subtasks: having agents first go into a line formation, and then switch to a circle formation. The formation control law from [48] was used to create decentralized consistent atoms containing controllers that drive agents into each of the formations separately. Using the GPS framework, a script written using the naive approach of executing the controllers for

achieving the two formations back to back was shown to not be executable. This was because the information flow graph of the system upon terminating the line formation controller did not allow for the circle formation controller to immediately start executing afterwards. However, the missing transition in the system’s information flow graph could be supplied by inserting a mode for nearest-neighbor averaging between the two existing modes. The resulting GPS was checked to be locally executable and a simulation was shown of agents executing the scripted sequence of decentralized controls to accomplish the original mission.

5.3 Generating Decentralized Consistent Atoms Using Optimal Decentralization

As shown in Figure 2 and demonstrated in the previous example, the effectiveness of GPS as a scripting tool relies on the richness of the decentralized consistent atom library that is available during the design phase. Up until now, decentralized consistent atoms have been created based on decentralized control laws that were taken from existing literature and performing additional analysis. However, this process of encapsulating decentralized consistent atoms is slow and each atom is created on a case-by-case basis only after carefully surveying related literature. Instead, a faster and more practical way to populate the atom library is to take a multi-agent task that is defined on the trajectory-level, and automatically generate both a decentralized controller and any additional information needed to make the corresponding decentralized consistent atom. Therefore, this section explains how to take the resulting sequences of decentralized controllers generated from the optimal decentralization algorithm in Chapter 4, and encapsulate it into decentralized consistent atoms to be stored in the atom library.

Continuing from the end of Section 4.2, assume that given some desired agent trajectory, a numerical optimization algorithm (e.g., steepest descent with Armijo stepsize) was used to find the mode parameters r_k^* and θ_k^* , as well as global switch times

τ_k^* , that yielded a tolerable final cost J . These optimized values define a decentralized control strategy for the multi-agent system to best track the desired motion with. We will now show how to encapsulate the generated sequence of decentralized controllers into a single decentralized consistent atom to be stored in the atom library.

Recall that in Section 4.2, each agent computed a control signal based on the sensed relative displacements between itself and its neighbors in the static directed network. Hence, let the information flow graph used in this situation be given by the graph inducing function $s_\sigma(x) = (\mathcal{N}, E_\sigma, w_\sigma(x))$, where E_σ is fixed and $w_\sigma(x)((i, j)) = x^i - x^j$ gives the relative displacement between neighboring agents, for all $(i, j) \in E_\sigma$. Define the set containing the graph inducing function s_σ as

$$\mathbb{S}_\sigma = \{s_\sigma\}. \quad (81)$$

With the optimized parameters r_k^* and θ_k^* for each mode, let the decentralized multi-agent controller $\mathcal{U}_\sigma^*(s, x, t)$ contain the entire optimized sequence of decentralized controllers resulting from the optimal decentralization algorithm (44), where

$$\mathcal{U}_\sigma^{*i}(s, x, t) = \begin{cases} - \sum_{j \in N(i)} r_{ijk}^* \text{Rot}(\theta_{ijk}^*) (x^i - x^j) & , \text{ for } t \in [T_0 + \tau_{k-1}^*, T_0 + \tau_k^*) \\ 0 & , \text{ otherwise,} \end{cases} \quad (82)$$

for $k = 1, \dots, m$. In the above expression, T_0 corresponds to the time at which that particular controller is first executed by agents in the system. Note that this controller assumes that agents have access to an accurate clock. Such an assumption was also used in the examples in Section 5.2, where agents used timer interrupts for switching between line and circle formation atoms.

To ensure that the controller in mode k executes through its entire required duration, we define the function $\mathcal{C}_\sigma^*(s, x, t)$ for agents to locally check when a controller's execution can be terminated as a timer interrupt:

$$\mathcal{C}_\sigma^*(s, x, t) = \begin{cases} 1 & , \text{ if } t \geq T_0 + T \\ 0 & , \text{ otherwise,} \end{cases} \quad (83)$$

where recall that T is the duration of the multi-agent motion that is being tracked.

Finally, we define the set of initial graphs \mathbb{G}_σ^* and final graphs \mathbb{H}_σ^* . These sets will contain the information flow graphs corresponding to the beginning and end of the multi-agent system's optimized state trajectory. Let $x^* : [0, T] \rightarrow \mathbb{R}^{2N}$ be the state trajectory resulting from using the optimized mode parameters r_k^* and θ_k^* , as well as the optimized global switching times τ_k^* , on the multi-agent system dynamics (50). The set of initial and final graphs are then given by

$$\mathbb{G}_\sigma^* = \{s_\sigma(x^*(0))\} \text{ and } \mathbb{H}_\sigma^* = \{s_\sigma(x^*(T))\}. \quad (84)$$

Putting all the components together, a decentralized consistent atom can be constructed to encapsulate the optimized sequence of decentralized controllers used for tracking a desired multi-agent motion.

Lemma 5.3.0.2. *The atom \mathcal{A}_σ^* , created by using the results from the optimal decentralization algorithm in Chapter 4:*

$$\mathcal{A}_\sigma^* = (\mathbb{S}_\sigma, \mathbb{F}_\mathcal{I}, \mathbb{G}_\sigma^*, \mathbb{H}_\sigma^*, \mathcal{U}_\sigma^*, \mathcal{C}_\sigma^*), \quad (85)$$

is a decentralized consistent atom.

Proof. The results of the optimal decentralization example in Section 4.2 are time-invariant and use only relative displacement information between agents. Hence, any offset to the initial conditions will simply generate an agent trajectory with that same offset maintained throughout. Similarly, any delay in the execution of these control laws will simply create the same resulting trajectory but delayed as well. Therefore, executing the generated sequence of decentralized controllers for any initial state with induced graph in \mathbb{G}_σ^* will guarantee that the state trajectory is in the set \mathbb{H}_σ^* , after executing for a duration of T . The function \mathcal{C}_σ^* allows for an agent in the network to detect when the controller has been executed for this duration and then terminate. \square

To summarize, this chapter presented the GPS framework as a way to script a sequence of decentralized controllers that could be consecutively executed by a multi-agent system. By encapsulating simple decentralized control laws into atoms, more complex multi-agent control strategies were made by stringing together multiple atoms and then optimized using mode sequencing and scheduling operations. In particular, the initial and terminating graph sets \mathbb{G} and \mathbb{H} within each atom allowed for sequences of controllers to be constructed where all network topological requirements for the controllers were maintained throughout execution. Examples were provided to illustrate how one can encapsulate existing decentralized control laws from literature into decentralized consistent atoms for usage with GPS. Moreover, it was shown how the optimal decentralization algorithm from Chapter 4 could generate decentralized consistent atoms that let agents track specific multi-agent motions with a static offset. Referring to Figure 2, these atom construction methods could then be used to populate an atom library that would expand the range of multi-agent decentralized control strategies that could be scripted using GPS.

CHAPTER VI

APPLICATIONS

The previous chapters presented a suite of tools for designing multi-agent systems. Together, these tools support different stages of the multi-agent design methodology presented in Chapter 1. This chapter will show how the underlying ideas behind each of the tools can be used in a variety of applications. To illustrate this, three examples will be presented in this chapter: air traffic merging and spacing, multi-UAV convoy protection, and an educational tool for robotics. Concepts from the tools that were developed in this dissertation, as well as the overall design methodology shown in Figure 2, have been used to produce elegant solutions for each of these three scenarios.

In each of the applications, the original task is first broken into a series of subtasks and a multi-agent system is selected or provided. After decentralized controllers have been chosen for agents to perform each of the subtasks with, the controllers are then strung together to form a more complex coordination strategy. When necessary, real-time monitoring of the environment is used to detect any changes in the operating conditions during execution and agents will react locally to these changes. It should be noted that the purpose of this chapter is not to go into the intricate details of each solution. Instead, the purpose is to showcase how concepts behind the tool suite that was developed in this dissertation have been successfully used in a variety of realistic and complex applications.

This chapter is organized as follows: Section 6.1 presents an example of using the presented design methodology to come up with a solution for merging and spacing air traffic during terminal approaches under the FAA's NextGen framework. Section 6.2 applies similar ideas towards solving a collaborative multi-UAV convoy protection

problem within a dynamic and hostile environment. Finally, Section 6.3 describes how those same concepts were used to create a final project for the graduate-level ECE8823 course in networked control systems that has been used annually since the Fall 2010 semester at the Georgia Institute of Technology.

6.1 Air Traffic Merging and Spacing

The first application to be considered is a solution for the merging and spacing of heterogeneous aircraft during terminal approaches. In particular, the scenario centers around aircraft flying on multiple legs of flight that must merge onto a single leg as shown in Figure 15. The goal is to make as little changes to the individual aircraft's speed or flight path deviation as possible, while still allowing for each aircraft to maintain at least a minimum specified distance away from other aircraft at all times. To make the scenario more realistic, the aircraft are assumed to be heterogeneous in that different aircraft require different separation distances from its neighbors. As discussed in the literature review in Section 1.2.5, existing solutions to this problem are mainly centralized and involve the air traffic controller telling each aircraft how to avoid conflicts on a case-by-case basis. The solution to be presented in this section is based off of the work in [23, 24, 120]. In particular, a distributed solution to the merging and spacing problem will be presented that makes use of the ADS-B communication protocol, which is part of the FAA's NextGen program. Through ADS-B, aircraft pass messages to one another and perform a dual decomposition to resolve merging conflicts in a distributed manner by negotiating for merging times that optimize a pairwise cost. Using a similar concept as the optimal decentralization algorithm from Chapter 4, local parameterized flight plans are then optimized to produce trajectories that ensure a safe and conflict-free merging of aircraft.

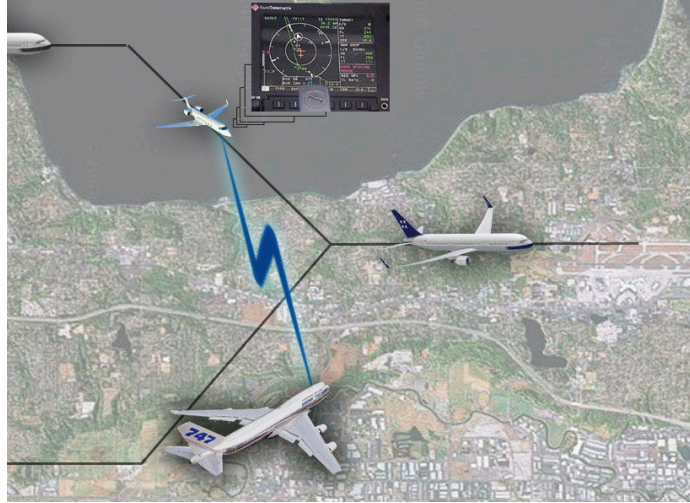


Figure 15: Aircraft on different legs of flight must merge while avoiding conflicts.

6.1.1 Problem Formulation

6.1.1.1 Task

To phrase the merging and spacing problem concisely, first assume that each aircraft has a unique ID, given by some positive integer. Let Ξ denote the set of all aircraft types and let $y : \mathbb{N} \rightarrow \Xi$ map each aircraft's ID to its associated aircraft type, i.e., the function $y(i)$ returns the type of Aircraft i . In this problem, we will assume that different types of aircraft are present, each with their own minimum separation requirements. Therefore, denote the spacing required for an aircraft of type k following an aircraft of any type as Δ_{III}^k , for each $k \in \Xi$. Finally, let the set of all inter-aircraft separations be given by $\mathcal{D} = \{\Delta_{\text{III}}^k \mid \forall k \in \Xi\}$. Referring to Figure 16, we will begin by looking at the problem of merging two legs of air traffic onto a single terminal leg, where each Aircraft i must maintain a spacing of at least $\Delta_{\text{III}}^{y(i)} \in \mathcal{D}$ with the aircraft merging in front of it at all times.

6.1.1.2 Subtasks

The merging and spacing procedure is divided into three phases that are marked by waypoints. These three phrases will be referred to as Phase I, the *Negotiation*

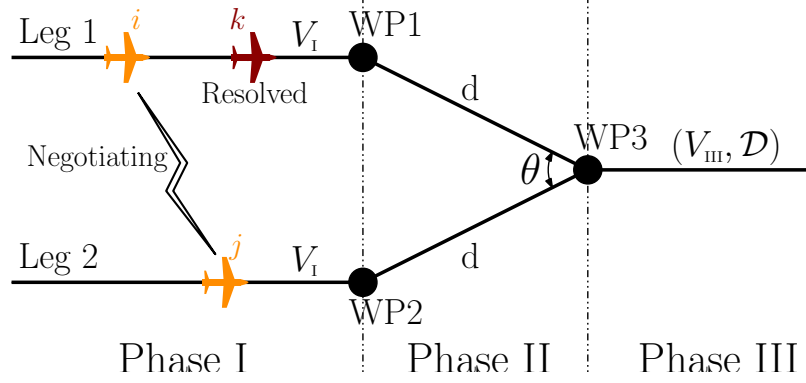


Figure 16: Top view of a two-track merging fork at the terminal phase of flight.

Phase; Phase II, the *Action Phase*; and Phase III, the *Terminal Approach Phase*. Referring to Figure 16, each aircraft will be given a sequence of timestamped waypoints to fly to along their leg of flight. Thus, an aircraft i flying on Leg 1 is given $\{(\text{WP1}, t_i^{\text{WP1}}), (\text{WP3}, t_i^{\text{WP3}})\}$, where t_i^{WP1} and t_i^{WP3} are the times at which aircraft i should be at WP1 and WP3, respectively. Similarly, an aircraft j flying on Leg 2 is given $\{(\text{WP2}, t_j^{\text{WP2}}), (\text{WP3}, t_j^{\text{WP3}})\}$.

6.1.1.3 Default Decentralized Control Strategy

In Phase I, the *Negotiation Phase*, aircraft approach waypoints WP1 and WP2 with a constant ground track speed V_I , spaced at least Δ_I apart from the aircraft in front of it on the same leg. During this approach, aircraft on opposing legs will conduct pairwise negotiations to determine arrival times at WP3 and flight plans over Phase II so as to maintain a safe separation with other aircraft. In Phase II, the *Action Phase*, each aircraft executes the negotiated flight plan to travel from WP1/WP2 to WP3. As seen in Figure 16, both WP1 and WP2 are assumed to be a distance d from WP3 at an angle θ apart, and the two dimensional problem is considered where tracks refer to the ground track of the aircraft.

The flight plan constitutes a ground track speed $V_{II} \in [V_{\min}, V_{\max}]$ and a path deviation $h \in [0, h_{\max}]$ from the straight line path between WP1/WP2 and WP3. As in [8, 37], changing V_{II} and h modifies the arrival time at WP3, which will be used

to space merging aircraft. This is illustrated in Figure 17. Note that path deviations occur in the direction opposite to the other leg of flight.

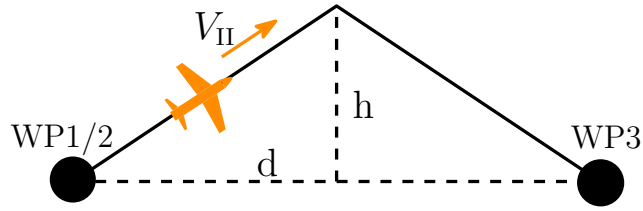


Figure 17: Overhead view of ground track speed and path deviations during Phase II.

In Phase III, the *Terminal Approach Phase*, each Aircraft i approaches the terminal with constant ground track speed V_{III} and must be at least $\Delta_{III}^{y(i)}$ away from the aircraft in front of it, as shown in Figure 18.

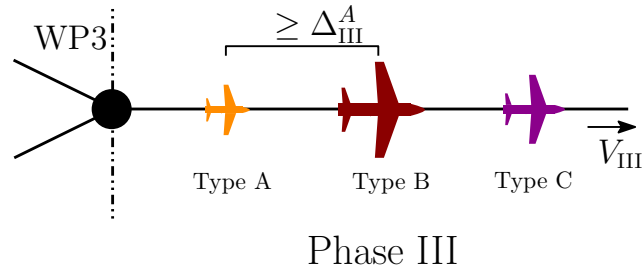


Figure 18: Ground track speed and aircraft separation during Phase III.

Throughout the three phases, aircraft are assumed to have access to the following global information as labeled in Figures 16 - 18: V_I and Δ_I are the ground track speed and minimum spacing that aircraft fly at in Phase I, d is the minimum distance required to fly in Phase II, V_{\min} and V_{\max} are respectively the minimum and maximum ground track speeds that aircraft can fly during Phase II, h_{\max} is the maximum allowable path deviation in Phase II, and V_{III} and \mathcal{D} are respectively the constant ground track speed of all aircraft and the set of minimum required separation by aircraft type during Phase III. The mapping of aircraft to aircraft type, y , is also known to all aircraft.

6.1.1.4 Real-Time Adaptation Strategy

To resolve potential merging conflicts, aircraft on opposing legs will negotiate during Phase I for arrival times at WP3 that ensures a safe separation. Assume that all aircraft initially start as being ‘unresolved’. The two unresolved aircraft that are closest to the merge point (one from each leg) will be the first pair to negotiate for an arrival time. After the pair has negotiated, the aircraft with the earliest arrival time will be assigned that arrival time and hence, be labeled the most recently resolved aircraft. The other aircraft, still unresolved, will then conduct pairwise negotiations with the next unresolved aircraft on its opposing leg for an arrival time at WP3. This arrival time should not only allow for the two aircraft to maintain a separation with each other, but also with the most recently resolved aircraft, when merging. Pairwise negotiation is continued in this manner until all merging aircraft on both legs of flight are assigned arrival times.

Let Aircraft i be the next unresolved aircraft on Leg 1 and Aircraft j be the next unresolved aircraft on the Leg 2, while Aircraft k is the most recently resolved aircraft. The following information is known to Aircraft i : $t_i^{\text{WP1/2}}$ is Aircraft i 's expected arrival time at WP1/WP2, $t_{i,0}^{\text{WP3}}$ is Aircraft i 's Estimated Time of Arrival (ETA) at WP3 if choosing $V_{\text{II}} = V_1$ and $h = 0$ in Phase II, τ_i is the set of Aircraft i 's *feasible* arrival times at WP3 while maintaining $\Delta_{\text{III}}^{y(i)}$ separation from Aircraft k in Phase III, and t_k^{WP3} is Aircraft k 's resolved time of arrival at WP3. Similarly, Aircraft j will know $t_j^{\text{WP1/2}}$, $t_{j,0}^{\text{WP3}}$, τ_j , and t_k^{WP3} . Aircraft i and j will also need to communicate additional information to each other throughout the negotiation process. These additional parameters are explained later in Section 6.1.3 as part of the proposed distributed solution.

6.1.1.5 Allowable Flight Plans for Phase II

A fixed arrival time at WP3 for any Aircraft i on either leg leads to a corresponding set of possible (V_{II}, h) pairs that can be chosen for Phase II to meet the arrival time. Vice versa, bounds on V_{II} and h limit which arrival times at WP3 can be achieved. The fastest that Aircraft i can plan to arrive at WP3 is when it flies in a straight line using the maximum ground track speed, corresponding to $V_{II} = V_{\max}$ and $h = 0$. Thus, T_{\min} , the soonest that Aircraft i can reach WP3, is given by

$$T_{\min} = t_i^{\text{WP1/2}} + \frac{d}{V_{\max}}. \quad (86)$$

The slowest that Aircraft i can reach WP3 is by flying at the minimum ground track speed with the greatest path deviation, corresponding to $V_{II} = V_{\min}$ and $h = h_{\max}$. As a consequence, T_{\max} , the latest that Aircraft i can reach WP3, is

$$T_{\max} = t_i^{\text{WP1/2}} + \frac{2}{V_{\min}} \sqrt{h_{\max}^2 + \frac{d^2}{4}}. \quad (87)$$

The set of *reachable arrival times* at WP3 for Aircraft i arriving at WP1/WP2 at time $t_i^{\text{WP1/2}}$ is therefore given by $R_i = [T_{\min}, T_{\max}]$. Let $\delta_{\text{III}}^i = \frac{\Delta_{\text{III}}^{y(i)}}{V_{\text{III}}}$ denote the amount of time that Aircraft i must arrive later than the most recently resolved aircraft at WP3 so as to ensure the appropriate Phase III spacing of $\Delta_{\text{III}}^{y(i)}$. Suppose Aircraft k is the most recently resolved aircraft with arrival time t_k^{WP3} at WP3, then let the set of *feasible arrival times* at WP3 for Aircraft i be $\tau_i = R_i \cap [t_k^{\text{WP3}} + \delta_{\text{III}}^i, \infty)$. Aircraft negotiating for arrival times must therefore choose from their respective sets of feasible arrival times.

Assuming that WP1/WP2 is a distance d from WP3 and it is desired to reach WP3 at time $t_i^{\text{WP3}} = t_i^{\text{WP1/2}} + T \in R_i$, it is possible to do so with any choice of $(V_{II}, h) \in \mathbb{S}(d, T)$ where

$$\mathbb{S}(d, T) = \left\{ (V, h) \in [V_{\min}, V_{\max}] \times [0, h_{\max}] \mid V = \frac{2}{T} \sqrt{h^2 + \frac{d^2}{4}} \right\}. \quad (88)$$

Having defined the set of allowable flight plans in Phase II, the next step is to develop a distributed negotiation procedure, along with a set of feasibility conditions, to determine terminal phase arrival times that maintain inter-aircraft separation. Furthermore, the negotiated arrival times must minimize pairwise aircraft costs. To do so, optimal local flight plan parameters for each aircraft will be computed in a way similar to that used in the optimal decentralization tool from Chapter 4.

6.1.2 Feasibility Conditions

Before discussing the negotiation aspect of this framework, we will identify a set of sufficient conditions on the geometry and operating conditions of merging forks for our algorithm to merge any combination of incoming aircraft (out of those which are specified initially), while ensuring that the minimum separation distance (relative to aircraft type) is maintained at all times. It is first shown that conditions exist on the interval length and intersections of the reachable time sets R_i , for all Aircraft i , such that aircraft on opposite legs performing a pairwise negotiation can agree on reachable arrival times at WP3 that guarantee a minimum separation between each other and also the previously resolved aircraft when in Phase III. This leads to conditions on the allowable choices of V_{\min} , V_{\max} , and h_{\max} on Phase II, which in turn gives conditions for choosing the ground track speed V_I and minimum aircraft spacing Δ_I for each leg during Phase I.

Define the length of a reachable time set $R_i = [a_i, b_i]$ as $|R_i| = |b_i - a_i|$. Denote the largest required inter-aircraft time separation by $\delta_{\text{III}}^{\max} = \frac{\max \mathcal{D}}{V_{\text{III}}}$. The first proposition will give conditions as to when it is always possible for a pair of airplanes on two different legs to find arrival times that ensure separation, irregardless of what arrival times previous aircraft had chosen.

Proposition 1. *If R_i , R_j , and R_{i+1} are such that $|R_x| \geq 2\delta_{III}^{max}$, for $x \in \{i, j, i + 1\}$, and $b_i \leq a_{i+1}$, then for all $c_i \in R_i$, there exists $c_j \in R_j$ and $c_{i+1} \in R_{i+1}$ such that $|c_i - c_j| \geq \delta_{III}^{max}$, $|c_i - c_{i+1}| \geq \delta_{III}^{max}$, and $|c_{i+1} - c_j| \geq \delta_{III}^{max}$.*

Proof. Choose

$$\begin{aligned}
 c_j &= a_j & \text{and} & & c_{i+1} &= b_{i+1}, & \text{if } a_j \leq c_i - \delta_{III}^{max} & \text{ or } a_j \leq a_{i+1} \\
 c_j &= c_i + \delta_{III}^{max} & \text{and} & & c_{i+1} &= b_{i+1}, & \text{if } c_i + \delta_{III}^{max} \in R_j \\
 c_j &= b_j & \text{and} & & c_{i+1} &= a_{i+1}, & \text{otherwise.}
 \end{aligned}$$

□

Suppose Aircraft i and $i + 1$ are on one leg and Aircraft j is on the opposite leg. The above proposition says that as long as certain conditions on the feasible time sets are met, any choice of arrival time at WP3 by Aircraft i has corresponding choices of arrival times at WP3 for Aircraft $i + 1$ and j such that the three maintain a separation of at least $\max \mathcal{D}$ from each other in Phase III. The maximum aircraft type separation is used here to ensure that all smaller spacings are accommodated. This result can be used to show that the proposed pairwise negotiation algorithm is guaranteed to result in arrival times for each aircraft that ensure separation in Phase III.

Theorem 6.1.2.1. *If the following conditions are satisfied for every Aircraft i and $i + 1$ following behind it on the same leg:*

$$R1 : |R_i| \geq 2\delta_{III}^{max}, \text{ where } |R_i| = \frac{2}{V_{min}} \sqrt{h_{max}^2 + \frac{d^2}{4}} - \frac{d}{V_{max}},$$

$$R2 : b_i \leq a_{i+1}, \text{ for } R_i = [a_i, b_i] \text{ and } R_{i+1} = [a_{i+1}, b_{i+1}],$$

then pairwise negotiation will allow all aircraft to agree on arrival times at WP3 that guarantee an inter-aircraft separation of least $\max \mathcal{D}$ in Phase III for all types of aircraft.

Proof. Suppose some Aircraft $i + 1$ and j are engaging in a pairwise negotiation, with a previously resolved Aircraft i (if one exists). Proposition 1 guarantees that independent of what arrival time t_i^{WP3} Aircraft i chose, there is a set of $(t_{i+1}^{\text{WP3}}, t_j^{\text{WP3}})$ pairs that allow for all three aircraft to maintain a separation of at least $\max \mathcal{D}$ in Phase III. Pairwise negotiation chooses a pair of arrival times for Aircraft $i + 1$ and j within that set that occur after t_i^{WP3} . Without loss of generality, assume that $t_j^{\text{WP3}} < t_{i+1}^{\text{WP3}}$. Aircraft j now becomes the next resolved aircraft, where t_j^{WP3} is chosen such that Aircraft j is guaranteed a separation of at least $\Delta_{\text{III}}^{y(j)}$ from all other previously resolved aircraft in Phase III. This process then continues inductively, where Aircraft $i + 1$ and $j + 1$ must perform pairwise negotiation to determine a $(t_{i+1}^{\text{WP3}}, t_{j+1}^{\text{WP3}})$ pair, and repeats until all aircraft have negotiated arrival times that guarantee the minimum separation requirement is met in Phase III. \square

Condition R2 requires aircraft on the same leg in Phase I to have reachable arrival time sets that overlap at most only at the boundary of the intervals. This condition can be transformed to equivalent conditions on spacing for incoming aircraft on Legs 1 and 2.

Theorem 6.1.2.2. *Condition R2 mentioned in Theorem 3.1 is equivalent to the distance Δ_I between any two consecutive aircraft on the same leg during Phase I being greater than or equal to $V_I |R_i|$, where $|R_i|$ is as given in Theorem 6.1.2.1.*

Proof. Assume at time t_0 , Aircraft i is a distance $x_{\text{WP1/2}} - x_i$ from WP1/WP2 and Aircraft $i+1$ is following behind at a distance $x_{\text{WP1/2}} - x_{i+1}$ from WP1/WP2. Therefore, the arrival times at WP1/WP2 are

$$\begin{aligned} t_i^{\text{WP1/2}} &= t_0 + \frac{x_{\text{WP1/2}} - x_i}{V_I}, \\ t_{i+1}^{\text{WP1/2}} &= t_0 + \frac{x_{\text{WP1/2}} - x_{i+1}}{V_I}. \end{aligned}$$

From Equations (86) and (87) and letting $R_i = [a_i, b_i]$ and $R_{i+1} = [a_{i+1}, b_{i+1}]$, it

follows that

$$b_i = t_i^{\text{WP1}/2} + \frac{2}{V_{\min}} \sqrt{h_{\max}^2 + \frac{d^2}{4}} \quad \text{and} \quad a_{i+1} = t_{i+1}^{\text{WP1}/2} + \frac{d}{V_{\max}}.$$

Substituting into Condition R2 results in

$$\Delta_I = x_i - x_{i+1} \geq V_I \left(\frac{2}{V_{\min}} \sqrt{h_{\max}^2 + \frac{d^2}{4}} - \frac{d}{V_{\max}} \right) = V_I |R_i|.$$

□

Sufficient conditions also exist that ensure aircraft on Phases I and II do not violate the minimum separation requirement, which are presented in the following theorem.

Theorem 6.1.2.3. *Assuming conditions R1 and R2 are met, a sufficient condition on θ , the angle between Legs 1 and 2, which guarantees that aircraft on Phase II maintain their required minimum separation is given by*

$$V_{\min} \cos \left(\frac{\theta}{2} \right) \geq V_{III} \quad \text{and} \quad \pi \geq \theta \geq \max\{\theta', \theta^*\},$$

where θ' and θ^* are given by

$$\begin{aligned} \alpha_1 \cos^2(\theta^*) + \alpha_2 \cos(\theta^*) + \alpha_3 &\geq 0, \quad \text{and} \\ d \sin \left(\frac{\theta'}{2} \right) &\geq \frac{\max \mathcal{D}}{2}, \end{aligned}$$

with $\alpha_1, \alpha_2, \alpha_3$ defined in (90), (91), (92) respectively.

Proof. For the purposes of analysis, let $t = 0$ be the time at which Aircraft 1 is at WP3 while Aircraft 2 is trailing behind and spaced as closely as possible at a distance of $\Delta_{III}^{y(2)}$. Tracing aircraft trajectories backward in time by defining $s = -t$, the minimum distance between two aircraft occurs when they do not deviate from the straight path, and where Aircraft 1 travels at V_{\max} while Aircraft 2 travels at V_{\min} in Phase II. As shown in Figure 19, the distance from Aircraft 1 to WP3 is $e_1(s) = V_{\max}s$

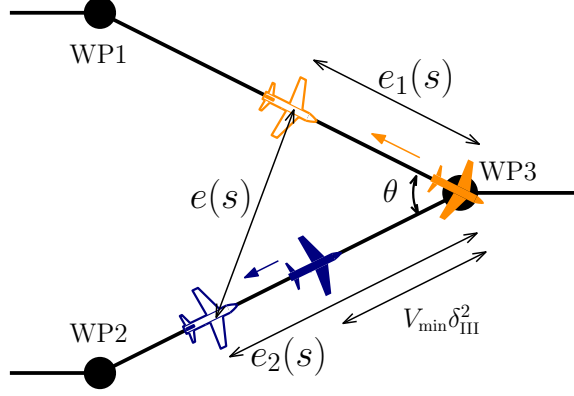


Figure 19: Diagram of Phase II used for proof of Theorem 3.3

and from Aircraft 2 to WP3 is $e_2(s) = V_{\min}(s + \delta_{\text{III}}^2)$, while the distance between the two Aircraft, $e(s)$, can be computed from the law of cosines. Solving for the time $s^* \geq 0$ when the minimum distance is achieved and making sure that $e(s^*) \geq \Delta_{\text{III}}^{y(2)}$, gives a condition on the minimum allowed inter-leg angle θ^* , such that

$$\alpha_1 \cos^2(\theta^*) + \alpha_2 \cos(\theta^*) + \alpha_3 \geq 0 \quad (89)$$

with

$$\alpha_1 = \frac{-V_{\max}^2 V_{\min}^2}{V_{\text{III}}^2}, \quad (90)$$

$$\alpha_2 = 2V_{\max} V_{\min}, \quad (91)$$

$$\alpha_3 = \frac{V_{\max}^2 V_{\min}^2}{V_{\text{III}}^2} - V_{\max}^2 - V_{\min}^2. \quad (92)$$

In addition, to ensure that aircraft on opposing legs maintain a separation from each other while in Phase I, Legs 1 and 2 must be at least $\max \mathcal{D}$ apart, meaning that $d \sin\left(\frac{\theta'}{2}\right) \geq \frac{\max \mathcal{D}}{2}$. Hence, it is required that angle $\theta \geq \max\{\theta', \theta^*\}$.

Finally, it must be checked that aircraft flying on Phase II and approaching the merge point maintain spacing with aircraft already flying in Phase III. In order to do this, let Aircraft 1 and 2 both be in Phase III where Aircraft 2 is at the merge point and is $\Delta_{\text{III}}^{y(2)}$ behind Aircraft 1. For purposes of analysis, let this time be $t = 0$. An expression for the inter-aircraft distance traced back in time as Aircraft 2 moves back

into Phase II, while Aircraft 1 remains in Phase III, is given by the following:

$$r^2(t) = (V_{II}t)^2 + (\Delta_{III}^{y(2)} - V_{III}t)^2 - 2V_{III}t(\Delta_{III}^{y(2)} - V_{III}t) \cos\left(\pi - \frac{\theta}{2}\right), \quad (93)$$

for all $t \in [-\delta_{III}^2, 0]$. In order to ensure spacing, it is required that $\min_{t \in [-\delta_{III}^2, 0]} r^2(t) \geq (\Delta_{III}^{y(2)})^2$. This function is clearly a quadratic of the form $at^2 + bt + c$, so to ensure concavity and that the minimum exists at $t = 0$ (where it is known that the two aircraft have adequate separation), it is required that $a \geq 0$ and $b \geq 0$. Therefore, the ‘b’ term gives the following condition:

$$2V_{II}\Delta_{III}^{y(2)} \cos\left(\frac{\theta}{2}\right) - 2\Delta_{III}^{y(2)}V_{III} \geq 0 \quad (94)$$

which implies that $V_{II} \cos\left(\frac{\theta}{2}\right) \geq V_{III}$. To ensure that any V_{II} chosen will satisfy the previous condition, it is required that

$$V_{\min} \cos\left(\frac{\theta}{2}\right) \geq V_{III}. \quad (95)$$

□

In summary, the following conditions are sufficient to guarantee complete feasibility:

$$\text{C1 } \Delta_I \geq V_I |R_i| \geq 2V_I \delta_{III}^{\max},$$

$$\text{C2 } V_{\min} \cos\left(\frac{\theta}{2}\right) \geq V_{III},$$

$$\text{C3 } \pi \geq \theta \geq \max\{\theta', \theta^*\}.$$

Having shown the conditions for which pairwise negotiation will ensure inter-aircraft separation throughout all three phases of flight, the original two-track merging fork will now be generalized to a binary tree that can merge an arbitrary number of legs of flight onto a single terminal leg.

6.1.2.1 Merging Multiple Legs of Flight

The proposed two-track merging fork, as shown in Figure 16, allows for air traffic from two separate legs to safely merge into one with guarantees that all aircraft will maintain a safe spacing from one another at all times. The feasibility results derived thus far can be used to generalize the two-track merging fork to allow for the merging of multiple legs of air traffic using a binary tree configuration as shown in Figure 20. In the figure, air traffic from legs 1 through 5 on the left all merge onto the terminal

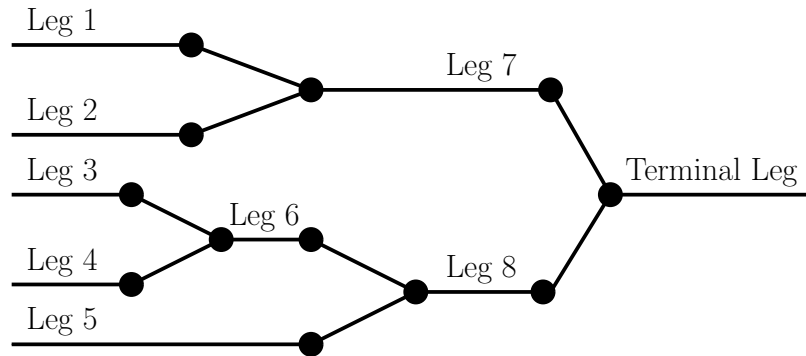


Figure 20: Binary tree structure for merging multiple tracks.

leg on the right, making use of intermediate legs 6, 7, and 8. The binary tree can be treated as a collection of two-track merging forks, where each leg in Phase I of a fork can be viewed as Phase III of another fork consisting of that leg and the two merging onto it. Thus, the speed and separation requirements on the terminal leg can be propagated backwards throughout the branches of the tree until feasible parameters for all legs have been determined.

As an example, let legs 7, 8, and the terminal leg of Figure 20 be Fork A, while legs 1, 2, and 7 form Fork B. The desired conditions \mathcal{D}_A and $V_{III,A}$ on the terminal leg will determine a range of options for choosing $\Delta_{I,A}$ and $V_{I,A}$ on legs 7 and 8 during the design of Fork A. However, leg 7 is both Phase I of Fork A and Phase III of Fork B, so for whatever $\Delta_{I,A}$ and $V_{I,A}$ values that are chosen, it is necessary for $\mathcal{D}_B = \{\Delta_{I,A}\}$ and $V_{III,B} = V_{I,A}$. With the conditions for Phase III of Fork A established, the feasibility

conditions can then be used to determine a range of valid choices of $\Delta_{I,B}$ and $V_{I,B}$ on legs 1 and 2.

It should be noted that the discussion above only addresses how to maintain a safe spacing amongst aircraft on the same fork. Additional care must be made in choosing the geometry of the fork (d and θ) so as to ensure that aircraft traveling on parallel forks, such as those on legs 2 and 3, are also able to maintain the necessary separations. Moreover, while a range of options for the ground track speed and inter-aircraft separation are available when designing a merging leg, it is expected that once chosen, all aircraft adhere to that one option when flying on the leg.

In order to avoid spacing conflicts between adjacent legs not on the same merging fork, a sufficient condition is presented based on the maximum path deviation regardless of the inter-leg angles for each fork. This condition is determined using the worst-case scenario where the two legs in question have parallel merging phases as in Figure 21. Referring to Figure 21, the adjacent fork spacing between Fork A (Legs 1, 2, and 7) and Fork B (Legs 3, 4, and 6), given by h_{AB} must allow for the maximum path deviation for each fork $h_{max,A}$ and $h_{max,B}$ as well as the safe spacing distance, $\max(\Delta_{I,A}, \Delta_{I,B})$. Therefore, $h_{AB} = h_{max,A} + h_{max,B} + \max(\Delta_{I,A}, \Delta_{I,B})$. Hence, any two adjacent forks, a and b , must be separated by

$$h_{ab} = h_{max,a} + h_{max,b} + \max(\Delta_{I,a}, \Delta_{I,b}). \quad (96)$$

6.1.3 Real-Time Adaptation Algorithm

6.1.3.1 Pairwise Optimization Problem

The pairwise negotiations for arrival times at WP3 will minimize a pairwise cost for both aircraft, consisting of the sum of Maneuvering and Delay costs for each aircraft and a joint Separation Cost. For an Aircraft i moving into Phase II, its Estimated Time of Arrival (ETA) at WP3, called $t_{i,0}^{WP3}$, is the time it takes to fly a straight line from WP1/WP2 to WP3 using the same ground track speed as in Phase I. Any

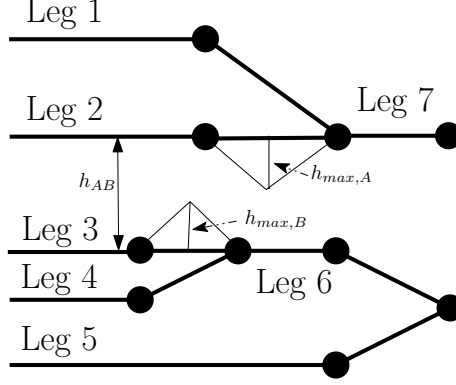


Figure 21: Zoomed in illustration of binary tree structure for merging multiple tracks showing the distance needed between adjacent legs to avoid conflicts amongst aircraft.

deviation in path, as well as changes in speed when switching between flight phases, correspond to an increase in fuel consumption and so is penalized.

Given an arrival time at WP3, the Maneuvering and Arrival Delay cost for an Aircraft i is

$$J_i(t_i^{\text{WP3}}) = \min_{(V_{\text{II}}, h)} (k_{1,i}h^2 + k_{2,i}((V_{\text{II}} - V_I)^2 + (V_{\text{III}} - V_{\text{II}})^2)) + k_{3,i}(t_i^{\text{WP3}} - t_{i,0}^{\text{WP3}})^2, \quad (97)$$

such that $(V_{\text{II}}, h) \in \mathbb{S}(d, t_i^{\text{WP3}} - t_i^{\text{WP1/2}})$. The weights $k_{1,i}, k_{2,i}, k_{3,i} \in \mathbb{R}_+$ may be chosen differently for each aircraft. Following the concept of generating optimal parameterized local controllers from the optimal decentralization algorithm in Chapter 4, the minimum term chooses the optimal V_{II} and h pair for aircraft i to arrive at WP3 at time t_i^{WP3} , while minimizing the penalty on deviations in path and ground track speed. In particular, $k_{1,i}$ penalizes path deviations, $k_{2,i}$ penalizes the changes in ground track speed when switching between flight phases, and $k_{3,i}$ penalizes the change in the aircraft's estimated time of arrival.

The Separation Cost penalizes a proposed pair of arrival times if they lead to aircraft having a separation greater than the minimum aircraft-specific separation in Phase III. The idea is to encourage aircraft to space themselves as closely as possible, without losing separation, so that later aircraft can have a wider range of feasible arrival times to choose from. This cost is referred to as a joint cost since it relies on

both t_i^{WP3} and t_j^{WP3} . Therefore, J_{ij} denotes the Separation cost if Aircraft i arrives first and is given by:

$$J_{ij}(t_j^{\text{WP3}}, t_i^{\text{WP3}}) = \gamma_{ij}(|t_j^{\text{WP3}} - t_i^{\text{WP3}}| - \delta_{\text{III}}^j)^2, \quad \gamma_{ij} > 0,$$

while $J(ji)$ denotes the Separation cost if Aircraft j arrives first:

$$J_{ji}(t_j^{\text{WP3}}, t_i^{\text{WP3}}) = \gamma_{ji}(|t_j^{\text{WP3}} - t_i^{\text{WP3}}| - \delta_{\text{III}}^i)^2, \quad \gamma_{ji} > 0.$$

Note that the desired separations, δ_{III}^i and δ_{III}^j , depend on who is the second to arrive at WP3 between the two negotiating aircraft.

There are two constraints on the allowable choices of WP3 arrival times. The first is that they must be feasible for the aircraft, and so it is required that $t_i^{\text{WP3}} \in \tau_i$, $t_j^{\text{WP3}} \in \tau_j$. The negotiated arrival times must also ensure that inter-aircraft separation, as determined by the type of the second aircraft to arrive, is achieved in Phase III. This is accomplished by the constraint $|t_j^{\text{WP3}} - t_i^{\text{WP3}}| \geq \delta_{\text{III}}^i$ when Aircraft j arrives first, and $|t_j^{\text{WP3}} - t_i^{\text{WP3}}| \geq \delta_{\text{III}}^j$ when Aircraft i arrives first.

Letting each Aircraft i and j be responsible for its own Maneuvering and Arrival Delay cost as well as half of the Separation Cost, the individual costs for each aircraft in the scenario when aircraft i arrives first are

$$\begin{aligned} U_i^i(t_i^{\text{WP3}}, t_j^{\text{WP3}}) &= J_i(t_i^{\text{WP3}}) + \frac{1}{2}J_{ij}(t_j^{\text{WP3}}, t_i^{\text{WP3}}), \\ U_j^i(t_i^{\text{WP3}}, t_j^{\text{WP3}}) &= J_j(t_j^{\text{WP3}}) + \frac{1}{2}J_{ij}(t_j^{\text{WP3}}, t_i^{\text{WP3}}), \end{aligned}$$

where the superscript i denotes that Aircraft i arrives first. If Aircraft j arrives first, the costs are

$$\begin{aligned} U_i^j(t_i^{\text{WP3}}, t_j^{\text{WP3}}) &= J_i(t_i^{\text{WP3}}) + \frac{1}{2}J_{ji}(t_j^{\text{WP3}}, t_i^{\text{WP3}}), \\ U_j^j(t_i^{\text{WP3}}, t_j^{\text{WP3}}) &= J_j(t_j^{\text{WP3}}) + \frac{1}{2}J_{ji}(t_j^{\text{WP3}}, t_i^{\text{WP3}}). \end{aligned}$$

Letting $a \in \{i, j\}$ denote which aircraft arrives first and $b \in \{i, j\}$ denote which aircraft arrives second, these costs can be combined to create the pairwise cost, and hence the following pairwise optimization problem:

Problem 6.1.1.

$$\min_{t_i^{\text{WP3}} \in \tau_i, t_j^{\text{WP3}} \in \tau_j} (U_i^a(t_i^{\text{WP3}}, t_j^{\text{WP3}}) + U_j^a(t_i^{\text{WP3}}, t_j^{\text{WP3}})),$$

such that $|t_j^{\text{WP3}} - t_i^{\text{WP3}}| \geq \delta_{\text{III}}^b$.

Note that a pair of negotiating aircraft must solve this problem twice, once for when Aircraft i arrives first and once for when Aircraft j arrives first. If both scenarios have valid solutions, then the two aircraft must decide who goes first by seeing which scenario produces the lowest pairwise cost. Next, a distributed pairwise negotiation will be used to solve this problem.

6.1.3.2 Distributed Solution using Dual Decomposition

Dual decomposition will be used by conflicting pairs of aircraft to reach agreement (as seen in [97]) on arrival times at WP3 that minimizes the pairwise cost between them, while satisfying the separation constraint. First, let t_{ij}^{WP3} be Aircraft i 's estimate of what Aircraft j 's arrival time at WP3 should be. The dual optimization problem to the primal Problem 6.1.1 is now written as

$$\max_{\lambda_1, \lambda_2} \min_{t_{ii}^{\text{WP3}}, t_{ij}^{\text{WP3}}} U_i^a(t_{ii}^{\text{WP3}}, t_{ij}^{\text{WP3}}) + U_j^a(t_{ji}^{\text{WP3}}, t_{jj}^{\text{WP3}}) + \lambda_1(t_{ii}^{\text{WP3}} - t_{ji}^{\text{WP3}}) + \lambda_2(t_{jj}^{\text{WP3}} - t_{ij}^{\text{WP3}}),$$

such that $t_{ii}^{\text{WP3}}, t_{ji}^{\text{WP3}} \in \tau_i$ and $t_{jj}^{\text{WP3}}, t_{ij}^{\text{WP3}} \in \tau_j$, with the constraints that

$$|t_{ij}^{\text{WP3}} - t_{ii}^{\text{WP3}}| \geq \delta_{\text{III}}^b \text{ and } |t_{jj}^{\text{WP3}} - t_{ji}^{\text{WP3}}| \geq \delta_{\text{III}}^b.$$

The primal problem has a bounded non-convex cost, meaning the dual problem has weak duality and so its solution cannot be guaranteed to result in a global minimum. Therefore, arrival times are sought after that achieve local minima for the pairwise constrained optimization problem.

In [86, 97], methods are presented for decomposing this dual optimization problem into subproblems that each aircraft can solve. As a result, the negotiation is broken

down into steps. First, each Aircraft solves a minimization problem based on its own arrival time estimates and given λ values. Then, arrival time estimates are communicated between the aircraft and each aircraft takes a gradient step to update its value of λ . Finally, the updated λ values are communicated to the other aircraft and the cycle begins again. These steps repeat until the other aircraft's suggested arrival time agrees with the aircraft's own calculated arrival time. The following describes the subproblems of the dual problem that are solved at each of these steps.

Aircraft i solves

$$\min_{t_{ii}^{\text{WP3}}, t_{ij}^{\text{WP3}}} U_i^a(t_{ii}^{\text{WP3}}, t_{ij}^{\text{WP3}}) + \lambda_1 t_{ii}^{\text{WP3}} - \lambda_2 t_{ij}^{\text{WP3}} \quad (98)$$

such that $t_{ii}^{\text{WP3}} \in \tau_i$, $t_{ij}^{\text{WP3}} \in \tau_j$, and $|t_{ij}^{\text{WP3}} - t_{ii}^{\text{WP3}}| \geq \delta_{\text{III}}^b$.

Aircraft j solves

$$\min_{t_{ji}^{\text{WP3}}, t_{jj}^{\text{WP3}}} U_j^a(t_{ji}^{\text{WP3}}, t_{jj}^{\text{WP3}}) - \lambda_1 t_{ji}^{\text{WP3}} + \lambda_2 t_{jj}^{\text{WP3}} \quad (99)$$

such that $t_{ji}^{\text{WP3}} \in \tau_i$, $t_{jj}^{\text{WP3}} \in \tau_j$, and $|t_{jj}^{\text{WP3}} - t_{ji}^{\text{WP3}}| \geq \delta_{\text{III}}^b$.

Next, Aircraft i and j take the gradient steps

$$\lambda_1^+ = \lambda_1 + t_{ii}^{\text{WP3}} - t_{ji}^{\text{WP3}}, \text{ and } \lambda_2^+ = \lambda_2 + t_{jj}^{\text{WP3}} - t_{ij}^{\text{WP3}}. \quad (100)$$

After the gradient step, the process repeats until an agreement on the arrival times is reached.

In order to solve these problems, Aircraft i must communicate t_{ij}^{WP3} and λ_1 to Aircraft j , while Aircraft j must communicate t_{ji}^{WP3} and λ_2 to Aircraft i . Such a communication can be done, for example, through the ADS-B communication protocol that is part of the FAA's NextGen program. Each aircraft must then solve the minimization problem once for the scenario when Aircraft i arrives first ($a = i, b = j$), and again for the scenario when Aircraft j arrives first ($a = j, b = i$). Afterwards, the negotiating aircraft will choose the best of the two scenarios (the one with the lowest pairwise cost) to execute.

6.1.4 Simulations

We will now showcase the performance of the proposed merging and pairwise negotiation protocol in a series of numerical simulations. The simulations show merging and spacing for a mixed fleet, where aircraft sizes are either “large” or “small”, each requiring a different spacing distance to ensure separation. The first simulation shows how each aircraft’s proposed arrival times converge throughout a pairwise negotiation, in the case when a large aircraft is negotiating with a small aircraft. The second simulation shows the proposed algorithm merging aircraft in a binary tree setting, where aircraft on three different legs of flight wish to merge onto a single terminal leg.

6.1.4.1 Pairwise Negotiation Simulation

The simulations start by demonstrating the convergence of each aircraft’s proposed arrival times during a pairwise negotiation. To show this, only the subtree of the binary tree in Figure 20, consisting of legs 7, 8, and the terminal leg is considered, which collectively is referred to as Fork A. The parameters for Fork A are:

$$\begin{aligned}
 V_{I,A} &= 1 & \Delta_{I,A} &= 12.6 \\
 V_{\min,A} &= 0.64 & V_{\max,A} &= 1.8 \\
 h_{\max,A} &= 2.9 & V_{III,A} &= 0.45 \\
 \Delta_{III,A}^{\text{Large}} &= 2.8 & \Delta_{III,A}^{\text{Small}} &= 1.5 \\
 d_A &= 10 & \theta_A &= \frac{\pi}{2},
 \end{aligned}$$

where $\mathcal{D}_A = \{\Delta_{III,A}^{\text{Large}}, \Delta_{III,A}^{\text{Small}}\}$. All the above parameters satisfy the derived feasibility conditions. Furthermore, since the air traffic consists of a mixed fleet, “large” aircraft require a larger separation, while “small” aircraft require a smaller separation, when following other aircraft.

The conversion from simulation units to physical units is given by:

$$\begin{aligned}
2.8 \text{ Distance Units} &= 2.5 \text{ NM} \\
2.8 \text{ Time Units} &= 1 \text{ minute} \\
1 \text{ Velocity Unit} &= 150 \text{ knots}
\end{aligned} \tag{101}$$

It is important to note that the above parameters were chosen merely to illustrate how one could design a merging fork for usage with the presented algorithm. However, no claim is being made as to how practical the merging fork and operating conditions used in this example would be in implementation.

Suppose that Aircraft 1, a “large” aircraft, and Aircraft 2, a “small” aircraft, are on opposing legs in Phase I, with no other aircraft preceding them. Aircraft 1 (large) is scheduled to arrive at WP1 at $t_1^{\text{WP1}} = 13$, while Aircraft 2 (small) is scheduled to arrive at WP2 at $t_2^{\text{WP2}} = 12$. Notice that if the two aircraft did not negotiate and just proceeded with their default flight plans of $V_{\text{II,A}} = V_{\text{I,A}}$ and $h = 0$, Aircraft 2 (small) would be the first to arrive at the merge point. However, the two aircraft would only have a separation of $V_{\text{III,A}} (t_1^{\text{WP1}} - t_2^{\text{WP2}}) = 0.45 < \Delta_{\text{III,A}}^{\text{Large}}$ in Phase III, and hence lose separation. Therefore, a pairwise negotiation is needed to resolve this merging conflict.

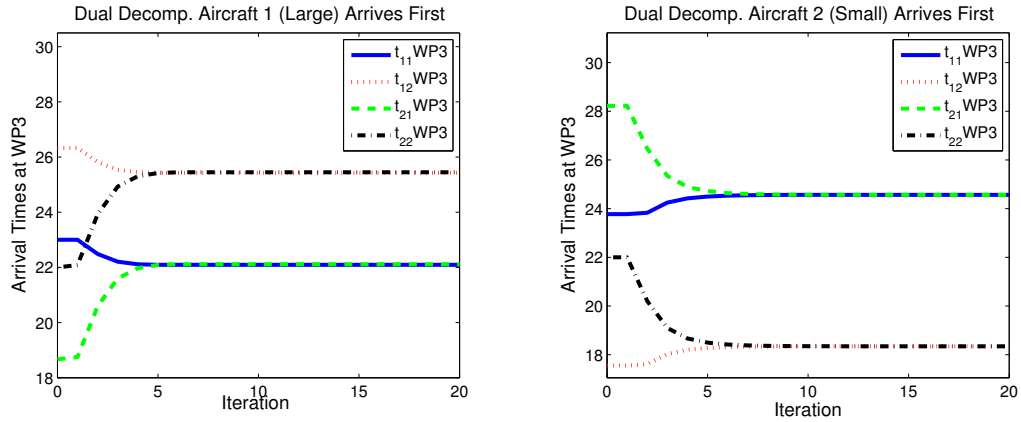
Based on the parameters of Fork A, the feasible time sets for Aircraft 1 and 2 to arrive at WP3 are $\tau_1 = [18.5556 \ 31.0629]$ and $\tau_2 = [17.5556 \ 30.0629]$, respectively. The goal of pairwise negotiation is to find a pair of feasible arrival times within the aircraft’s feasible time sets that ensure separation. If Aircraft 1 (large) is chosen to go first, the distance between the two aircraft must be at least $\Delta_{\text{III,A}}^{\text{Small}}$ upon arriving at WP3. Since the ground track speed in Phase III is constant, maintaining separation is equivalent to the arrival times being at least $\delta_{\text{III,A}}^2 = \frac{\Delta_{\text{III,A}}^{\text{Small}}}{V_{\text{III,A}}} = 3.3333$ apart. Alternatively, if Aircraft 2 (small) is chosen to go first, the separation between the aircraft must be at least $\Delta_{\text{III,A}}^{\text{Large}}$ upon arriving at WP3, which is the same as the arrival times

differing by at least $\delta_{\text{III,A}}^1 = \frac{\Delta_{\text{III,A}}^{\text{Large}}}{V_{\text{III,A}}} = 6.2222$.

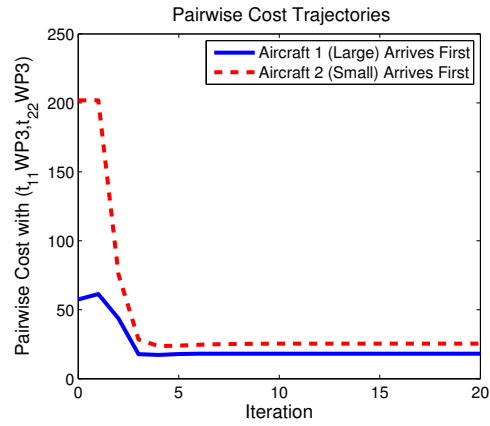
The pairwise negotiation cost weights for Aircraft 1 (large) are $k_{1,1} = 3$, $k_{2,1} = 8$, and $k_{3,1} = 3$, while the weights for Aircraft 2 (small) are $k_{1,2} = 10$, $k_{2,2} = 2$, and $k_{3,2} = 1$. Recall that k_1 penalizes any deviations in the flight path, k_2 penalizes changes in the ground track speed, and k_3 penalizes deviations from an aircraft's ETA (had it chosen $V_{\text{II,A}} = V_{\text{I,A}}$ and $h = 0$ for Phase II) at the merge point. The values of the weights thus define the preferences of each aircraft. If it is necessary to delay the arrival time at the merge point, Aircraft 1 would rather deviate its path than change its ground track speed, as seen by it weighing changes in ground track speed more in its cost. Aircraft 2, on the other hand, has opposite preferences and would rather change its ground track speed than deviate its path, in order to stall for time. Furthermore, since the k_3 term is larger for Aircraft 1, it wishes to arrive at WP3 at its ETA more than Aircraft 2. The weight in the joint cost was chosen to be $\gamma = 10$ to give some incentive for the two aircraft to space themselves as closely as possible without losing separation.

The results of performing a pairwise negotiation by running a dual decomposition for 20 iterations between the two aircraft are shown in Figures 22(a) and 22(b). Each iteration in the plots correspond to an exchange of information between the two negotiating aircraft. In both cases, the pairwise negotiations converge in that $|t_{11}^{\text{WP3}} - t_{21}^{\text{WP3}}| \rightarrow 0$ and $|t_{22}^{\text{WP3}} - t_{12}^{\text{WP3}}| \rightarrow 0$ as the number of iterations increase, i.e., both Aircraft 1 and 2 eventually agree on what Aircraft 1 should do during Phase II, and vice versa.

First, consider the case when Aircraft 1 (large) is chosen to go first. Figure 22(a) shows that the final negotiated arrival times are $t_{11}^{\text{WP3}} = 22.0918$ and $t_{22}^{\text{WP3}} = 25.4471$. Notice that the two negotiated arrival times ensure separation because they differ by 3.3553, which is greater than the required $\delta_{\text{III,A}}^2 = 3.333$. Next, in Figure 22(b), the case when Aircraft 2 (small) is chosen to go first results in final negotiated arrival



(a) Convergence of proposed arrival times when Aircraft 1 (large) goes first. (b) Convergence of proposed arrival times when Aircraft 2 (small) goes first.



(c) Pairwise cost trajectory using negotiated arrival times per iteration.

Figure 22: Arrival time agreement and pairwise cost minimization.

times $t_{11}^{\text{WP3}} = 24.5681$ and $t_{22}^{\text{WP3}} = 18.3459$. Once again, the two negotiated arrival times ensure separation since they differ by 6.2222, which exactly equals the required $\delta_{\text{III,A}}^1 = 6.2222$. Since both negotiations resulted in arrival times that would ensure a successful merging with separation maintained throughout Phase III, it is necessary to look at the final pairwise costs to determine which aircraft ultimately should go first.

The pairwise cost trajectories for both scenarios are shown in Figure 22(c). Note that the pairwise costs do not necessarily need to be monotonically decreasing throughout the negotiation since forcing t_{11}^{WP3} and t_{22}^{WP3} to satisfy the necessary spacing constraints, when they originally do not, may increase the cost. At the end of the negotiation, the case when Aircraft 1 (large) arrives first results in a final pairwise cost of $J = 18.1243$, while the case when Aircraft 2 (small) arrives first results in $J = 25.4488$. Upon evaluating the final pairwise costs for both of the valid scenarios, the aircraft decide amongst themselves that it is best for Aircraft 1 (large) to arrive at the waypoint first. Thus, Aircraft 1 is marked as being resolved and is scheduled to take the merge point first. It does so by choosing the optimal parameters (V_{II}, h) for its local controller in Phase II that will allow it to reach WP3 at the negotiated time of $t_{11}^{\text{WP3}} = 22.0918$, which is $V_{\text{II}} = 1.1201$ and $h = 0$. The negotiated flight plan for Aircraft 1 corresponds to an increase in its ground track speed during Phase II with no path deviation, in order to get to the merge point earlier than its original ETA. Since Aircraft 2 is still unresolved, it must now negotiate with the next unresolved aircraft behind Aircraft 1 for an arrival time at the merge point.

6.1.4.2 Binary Tree Simulation

Having demonstrated a pairwise negotiation in the previous simulation, a second simulation is presented that uses the pairwise negotiation protocol to merge three different legs of air traffic onto a single terminal leg. The three legs of air traffic will

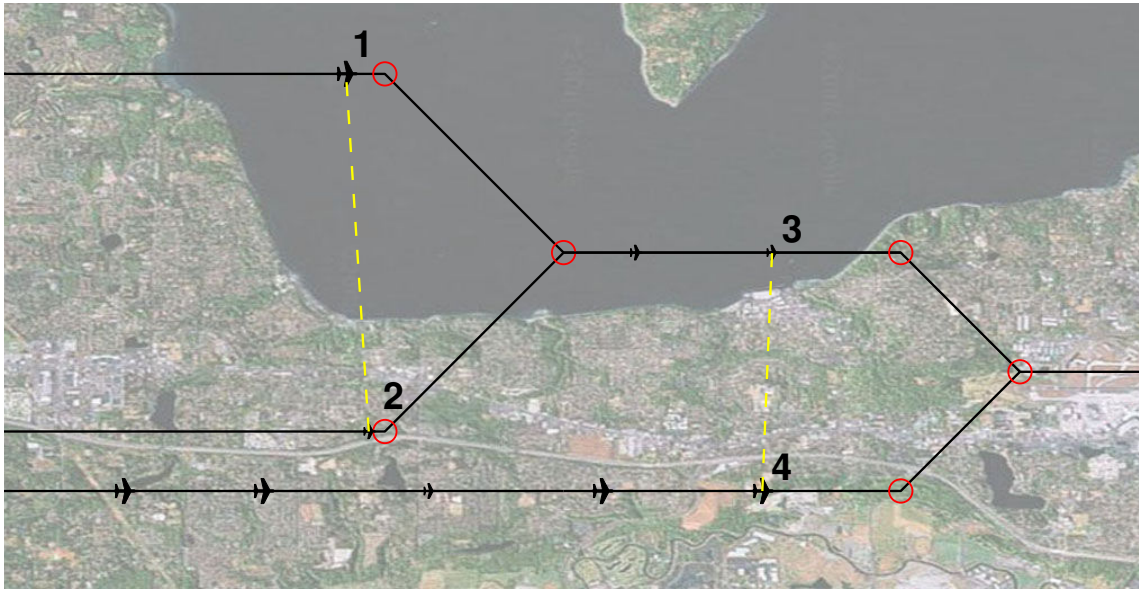
be composed of 2 two-track merging forks: Fork A from the previous simulation, and Fork B, composed of legs 1, 2, and 7 from Figure 20, whose parameters are given by

$$\begin{aligned}
 V_{I,B} &= 1 & \Delta_{I,B} &= 26 \\
 V_{\min,B} &= 1.42 & V_{\max,B} &= 10 \\
 h_{\max,B} &= 18 & V_{III,B} &= 1 \\
 \Delta_{III,B}^{\text{Large}} &= 12.6 & \Delta_{III,B}^{\text{Small}} &= 12.6 \\
 d_B &= 15 & \theta_B &= \frac{\pi}{2},
 \end{aligned}$$

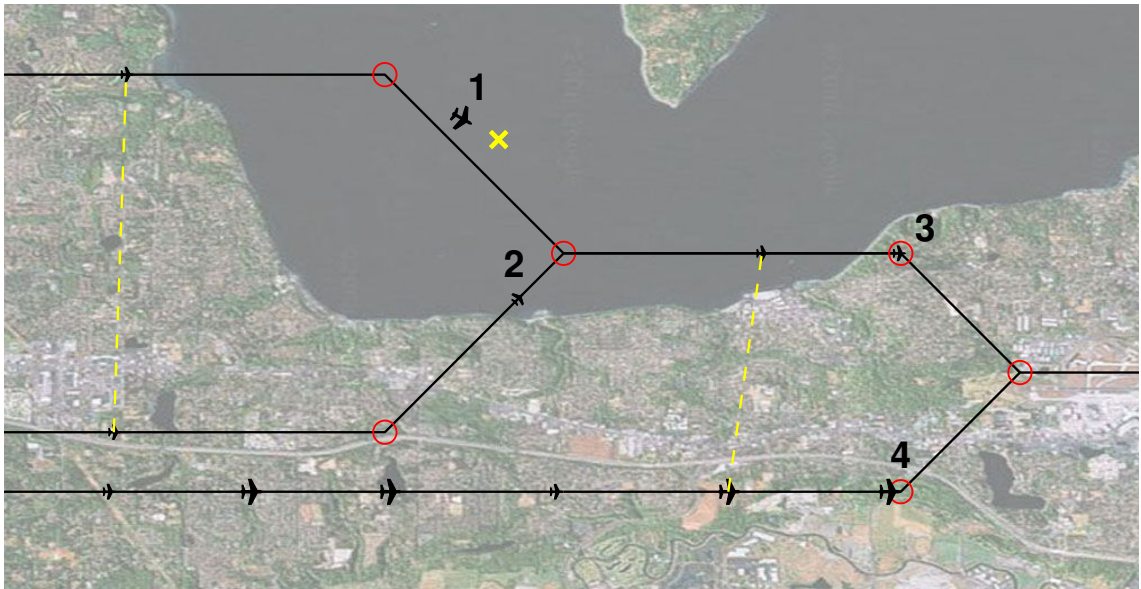
where $\mathcal{D}_B = \{\Delta_{III,B}^{\text{Large}}, \Delta_{III,B}^{\text{Small}}\}$. The set of parameters for Fork B also satisfy the feasibility conditions. Recall that when using multiple forks to create a binary tree, the parameters for Phase I of a fork are the same as the parameters that define Phase III of the fork preceding it. Therefore, special care was taken to ensure that $\Delta_{III,B}^{\text{Large}} = \Delta_{III,B}^{\text{Small}} = \Delta_{I,A}$ and $V_{III,B} = V_{I,A}$.

Since the parameters for each fork were chosen to satisfy the derived feasibility conditions, using the proposed pairwise negotiation protocol amongst merging aircraft will guarantee that aircraft will maintain a separation from each other at all times. The simulation of the binary tree was performed with incoming aircraft randomly inserted into leg 8 of Fork A with at least a separation of $\Delta_{I,B}$, and legs 1 and 2 of Fork B with at least a separation of $\Delta_{I,A}$, from all other aircraft on the same leg. Screenshots from the simulation showing how pairwise negotiation successfully merges the three legs of air traffic onto a single terminal leg are shown in Figure 23 and 24.

Although there are many aircraft seen in the simulation, only the actions taken by Aircraft 1 through 4, as marked accordingly in the figures are considered. Aircraft 1 and 3 are “large”, while Aircraft 2 and 4 are “small” aircraft. In Figure 23(a), both Aircraft 1 and 2 are approaching the merge point in Fork B. Similarly, Aircraft 3 and 4 are approaching Fork A’s merge point. Since both pairs of aircraft are reaching

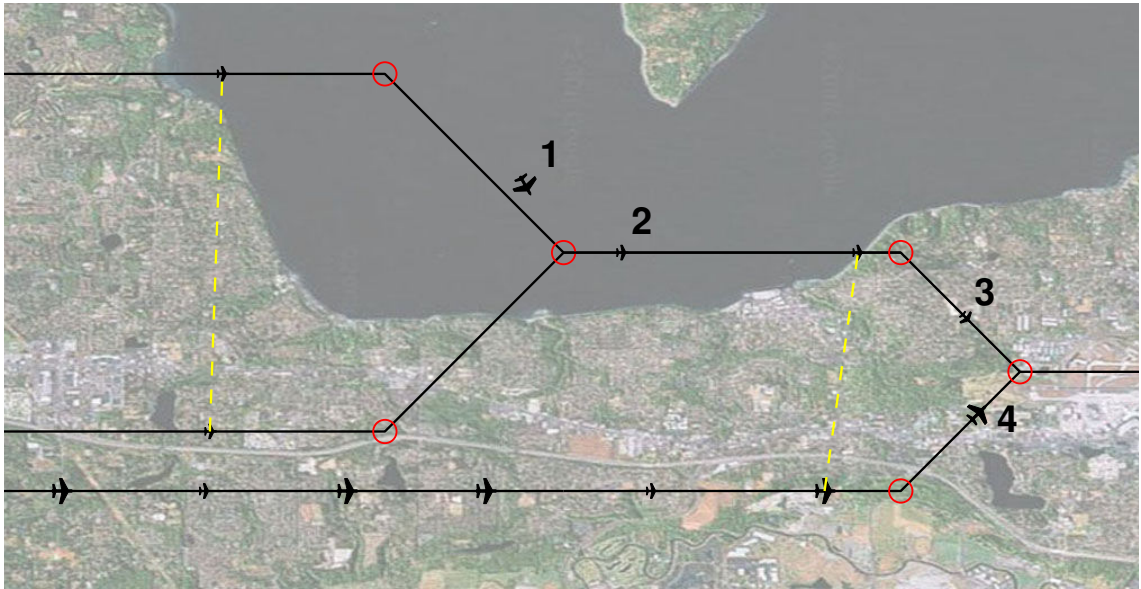


(a) Aircraft 1 and 2 approach the merge point in Fork B, Aircraft 3 and 4 similarly approach the merge point in Fork A.

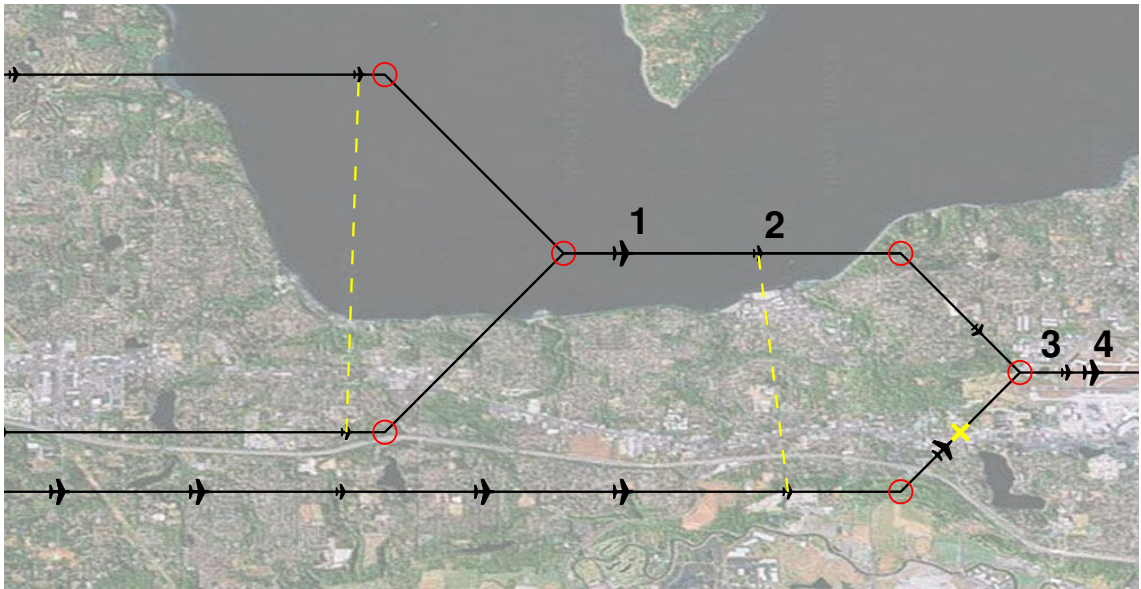


(b) Aircraft 1 yields to Aircraft 2 in Fork B by deviating its path. Aircraft 3 and 4 continue to negotiate in Fork A.

Figure 23: Part one of a simulated binary tree structure merging three legs of air traffic onto a single terminal leg.



(a) Aircraft 2 has merged in Fork B, Aircraft 1 follows behind at a safe distance. Aircraft 4 speeds up to take the merge point before Aircraft 3 in Fork A.



(b) Both Aircraft 1 and 2 have merged in Fork B and are now on same leg in Fork A. Aircraft 3 and 4 successfully merged in Fork A onto the terminal leg.

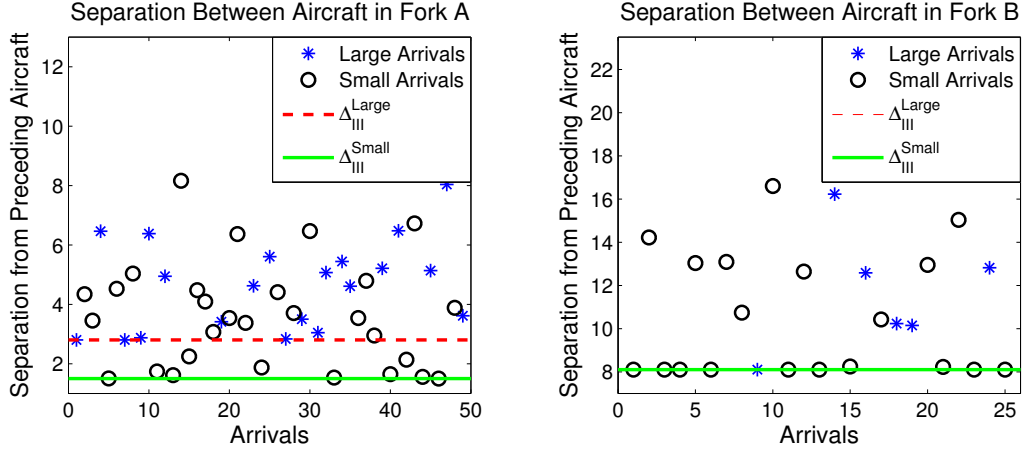
Figure 24: Part two of a simulated binary tree structure merging three legs of air traffic onto a single terminal leg.

WP1/2 of their respective forks at almost the same time, they will most likely lose separation if they continue using their default flight plans of $V_{II} = V_I$ and $h = 0$ during Phase II. Thus, to resolve these merging conflicts, both pairs of aircraft must perform pairwise negotiations and act accordingly if they wish to maintain a safe separation when merging.

Figure 23(b) shows that Aircraft 1 and 2’s negotiation resulted in Aircraft 1 taking a path deviation to delay its arrival time at the merge point. Figure 24(a) shows that Aircraft 3 and 4’s negotiation, on the other hand, has determined that the best course of action was for Aircraft 4 to increase its ground track speed, while Aircraft 3 decreases its ground track speed. Figure 24(b) shows that both pairs of aircraft have merged successfully and have maintained a safe separation with other aircraft. Aircraft 1 and 2 have merged onto the same leg in Phase I of Fork A and must now negotiate with aircraft on the opposing leg of Fork A to determine how to engage the next merge point. Aircraft 3 and 4 have both merged onto the terminal leg and can proceed to land in the terminal.

To verify concretely that the pairwise negotiations in the preceding simulation have succeeded in maintaining separation amongst aircraft, a plot of inter-aircraft spacing at the merge point of Fork A is shown in Figure 25(a), while a similar plot for the merge point in Fork B is shown in Figure 25(b). Looking at the plots, each arrival of a “large” aircraft on a fork has a separation of at least Δ_{III}^{Large} for that fork, and similarly all arrivals of “small” aircraft have a separation of at least Δ_{III}^{Small} for that fork. Therefore, the simulation confirms that pairwise negotiation was successful in safely merging aircraft from the three incoming legs onto a single terminal leg.

This section presented a distributed framework for merging and spacing heterogeneous aircraft during a terminal approach. By looking at the problem from the perspective of the design methodology in Figure 2, the merging and spacing task was partitioned into a sequence of subtasks involving aircraft having to move between a



(a) Spacing between consecutive aircraft arriving at the merge point for Fork A. (b) Spacing between consecutive aircraft arriving at the merge point for Fork B.

Figure 25: Plots of separation distances amongst consecutive aircraft arrivals for each two-track merging fork’s merge point in the binary tree merging simulation.

set of timestamped waypoints. A set of feasibility conditions were proven that gave conditions on both the geometry of the merging fork, and the operating conditions, so that merging conflicts could be avoided when using this framework. Under those conditions, aircraft were able to implement real-time adaptation by negotiating with potentially conflicting aircraft and modifying their flight plans accordingly. Using a similar concept as the optimal decentralization algorithm from Chapter 4, local parameterized flight plans were then optimized to produce trajectories that ensured a safe and conflict-free merging of aircraft. Finally, the framework was showcased in a series of simulations involving merging air traffic on multiple legs of flight into one.

6.2 Collaborative Multi-UAV Convoy Protection

The second application to be showcased is a collaborative multi-UAV convoy protection solution for a team of UGVs traveling in a dynamic and hostile environment. As illustrated in Figure 26, a team of UAVs must track and monitor a team of UGVs as it travels along a preplanned path. However, along the way are pop-up threats which the convoy must avoid getting too close to. The UAVs must work together to clear the threat or if it cannot be cleared, signal to the UGVs the need to replan the

path in order to avoid the threat. In particular, the problem of dispatching UAVs to threats must be solved here. As mentioned in the literature review in Section 1.2.6, one way to pose this problem involves having to solve an instance of the NP-hard TSP problem. However, in this section, we present work from [119] that takes a different view on the convoy protection problem and avoids the computational intractability all together.

With the use of some simplifying but realistic assumptions on the scenario, the problem reduces to one of assigning multiple UAVs to a single threat at a time. Because each UAV is capable of performing the same tasks, the assignment follows the same underlying principle presented in Chapter 3 on the controllability of homogeneous agents. Here, the agents refer to the UAVs while the target point consists of locations above the threat and above the convoy that is to be protected. The resulting multi-agent system must then be controlled to a permutation of this target point. However, in this case, the focus is not on controlling agents to get as close to the target points as possible. Instead, for practical purposes, a decentralized assignment algorithm is presented so as to perform the UAV-to-target assignment in a way that balances the UAVs' fuel consumption, thereby maximizing the time duration between having to refuel.

6.2.1 Problem Formulation

6.2.1.1 Task

A collection of UAVs must provide protection to a convoy of UGVs in the sense that (1) at least one UAV should always be placed above the convoy so as to provide protection from potential immediate threats, and (2) pop-up threats along the path traversed by the convoy must be visited and cleared by UAVs before the convoy arrives, which implies that UAVs must be dispatched away from the convoy to handle these threats. To make the assignment of UAVs to threats computationally tractable, only the closest uncleared threat to the convoy will be considered at any given moment.

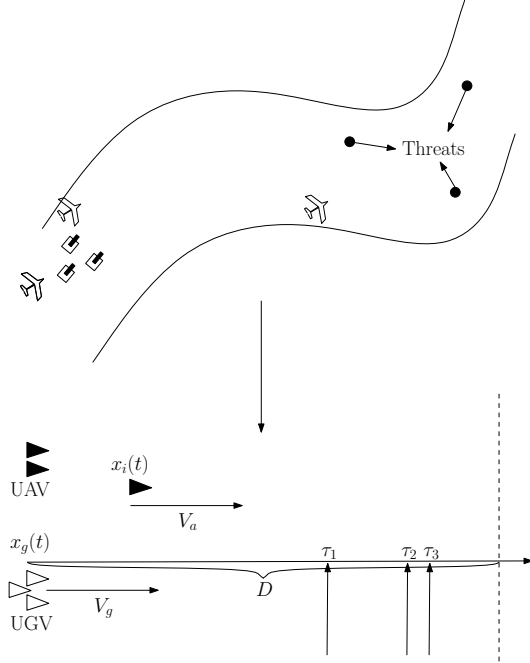


Figure 26: The multi-UAV convoy protection scenario with pop-up threats.

Moreover, the assignment must be done in such a way that the fuel consumption is balanced between the UAVs so that the time in-between having to refuel is maximized. To perform this assignment, the UAVs are able to directly communicate with one another. However, since they are operating in a hostile environment, the designation of a leader UAV to perform this assignment is not appropriate. In particular, the performance of the entire UAV team could potentially be compromised if the leader UAV becomes disabled or loses communication with the others. Therefore, the assignment of UAVs to threats must be done in a decentralized manner.

6.2.1.2 Subtasks

The UGVs must follow along the predetermined path which is given by a sequence of breadcrumb waypoints. A spline is then generated using these waypoints to create a smooth path for the convoy to travel. The UAVs, on the other hand, must hover closely above the convoy to monitor for any immediate threats. It is required that at least one UAV be above the convoy at all times. Of course, it is expected that

the UAVs also employ collision-avoidance tactics to maintain a safe distance with one another at all times.

6.2.1.3 Real-Time Adaptation Strategy

As soon as a single UAV detects a threat, the UAVs must perform a decentralized negotiation amongst themselves to determine which UAV is assigned to fly and hover above the threat for a fixed period of time in order to attempt to clear it, while ensuring that the fuel consumption is balanced amongst the UAVs. If the threat cannot be cleared, the convoy is informed by the UAVs so that it can replan its route by adding a deviation in its planned trajectory. Next, the technical details behind the decentralized fuel-balancing assignment algorithm used by the UAVs will be presented.

6.2.2 Real-Time Adaptation Algorithm

6.2.2.1 Fuel-Balancing UAV Assignment Problem Formulation

In order to formulate the decentralized fuel-balancing UAV assignment problem, we model the UGVs as a single point mass moving along a one-dimensional (not necessarily straight) path as shown in Figure 26. The UAVs are either flying alongside the UGVs (meaning they are at the same position along the path), or they can fly ahead of the UGVs. As they fly ahead of the UGVs, they do not have to follow the path. Instead, we will assume that they follow the Euclidean shortest distance to the target location. Hence, for this model, the motion of the UGVs is viewed as moving along a one-dimensional corridor which allows us to establish an order among the threats and significantly cuts down on the complexity of the problem. At the same time, this is a realistic assumption that does not significantly limit the applicability of the proposed method.

With this model, we do not consider the kinematic constraints of the UAVs and UGVs, and the UAVs are assumed to be able to change direction and turn around

while checking threats. At time t , the position of the UGV team is denoted to be $x_g(t)$. The positions of the individual UAVs are denoted as $x_i(t)$, where $i = \{1, 2, 3\}$ labels the UAVs.

We assume that the team can detect threats for a distance of D in front of the UGVs. Hence only threats contained in this window are known to the UGV-UAV team. At time t , we assume that there are $N(t)$ threats in this range. The location of threats are assumed to be fixed, and they are denoted as $\tau_j, j = 1, 2, \dots, N(t)$. The sequence of the known threats at time t are denoted as $\bar{\tau}(t) = [\tau_1, \tau_2, \dots, \tau_{N(t)}]$. This known threat sequence has time-varying length $N(t)$. $N(t)$ increases when a pop-up threat is detected, and decreases when a threat is cleared. Note that all threats (pop-up or not) are in front of the UGVs on the intended path.

Furthermore, we denote the total number of threats as N (where we have, for notational convenience, suppressed the explicit dependence on t , as will be done throughout), and the sequence of all threats is denoted as $\bar{\tau} = [\tau_1, \tau_2, \dots, \tau_N]$. The overall mission of the UAVs is to clear all the threats (assuming none of them are hostile). If any of the threats are hostile, then the path taken by the UGVs is re-planned, the problem is re-initialized, and the assignment algorithm is restarted.

Naturally, when the UAV is checking a threat, it should fly with a higher speed than when it is cruising along with the UGVs. The velocity of the UGV team is assumed to be V_g . We assume that when the UAVs are flying alongside the UGVs, they fly with the same speed V_g . However, when they are assigned to clear a threat or coming back after clearing a threat, they fly with the speed V_a . Throughout this section, we assume that $V_a > V_g$.

We denote the starting time of the overall mission as t_0 , and the time when all N threats are cleared as T . It should be noted that since we aim at developing a real-time algorithm to select and assign the UAVs to clear the threats, the only information available at time t is the $N(t)$ threats within range D , and our assignment algorithm

only uses this information to make decisions.

The problem considered here is to devise an algorithm to dispatch the UAVs and clear all upcoming threats, while balancing the fuel consumptions of the UAVs. The fuel consumption of i th UAV at time t is denoted as $f_i(t)$. $f_i(t)$ is considered to be the state of the UAVs, and they are known. It should be noted that at the beginning of the mission, the fuel consumption of all the UAVs are 0, hence $f_i(t_0) = 0, \forall i$. The details of how fuel consumption is calculated for the UAVs will be presented shortly. With this information, we are ready to formulate the UAV assignment problem.

Problem 6.2.2.1. *Design an algorithm to assign UAVs such that the maximum fuel consumption amongst the UAVs is minimized when all threats $\bar{\tau}$ are cleared. Hence, we aim to solve the following optimization problem:*

$$\min_{\Pi} \max_{j \in \{1,2,3\}} \{f_j(T)\}, \quad (102)$$

where Π is an assignment algorithm that maps from time to $\{1, 2, 3\}$.

Note that even though we only consider three UAVs in the problem formulation, the algorithms and results presented in this document can be extended to any number of UAVs.

6.2.2.2 UAV Fuel Model

The fuel consumption rate for the UAVs at any given moment will be a nonlinear function of its speed, weight, and altitude. Therefore, the fuel consumption rate of the i th UAV at time t is given by

$$\frac{df_i(t)}{dt} = Q(v(t), w(t), h(t)), \quad (103)$$

where $v(t), w(t), h(t)$ are the UAV's speed, payload weight, and altitude at time t , respectively.

In our convoy protection scenario, the UAVs will be assumed to only fly at two speeds: V_g when following the UGV convoy, and V_a when surveying and clearing

a threat. The payload weight and altitude of the UAVs are assumed to remain constant $w(t) = w^*$ and $h(t) = h^*$ throughout the individual maneuvers needed to inspect threats. Therefore, the fuel consumption rate of the i th UAV at time t can be simplified to

$$\frac{df_i(t)}{dt} = \begin{cases} Q(V_g, w^*, h^*) & , \text{ if } v(t) = V_g \text{ (UAV } i \text{ is following convoy),} \\ Q(V_a, w^*, h^*) & , \text{ if } v(t) = V_a \text{ (UAV } i \text{ is surveying/clearing threat).} \end{cases} \quad (104)$$

6.2.2.3 Selection Policy

In our framework, the selection policy is a function that maps time t to the set of the UAVs ($\{1, 2, 3\}$ in our case), and it is the solution to the optimization problem. This optimization policy is based on computing the fuel consumption associated with letting a UAV clear the target threat τ and return to the convoy. Even though this may actually not be what is done (as new threats are constantly reconsidered), the resulting algorithm minimizes the accumulated fuel consumption over all other assignments provided that the assigned UAV will return to the convoy after the threat is inspected. We take into account that the UGVs are moving constantly along the path with speed V_g , and that the closest threat τ may be a pop-up threat.

When an event triggers at time t , the following optimization problem is solved:

$$P_1(t) = \arg \min_{i \in \{1, 2, 3\}} \max_{j \in \{1, 2, 3\}} f_j(t) + \begin{cases} Q(V_a, w^*, h^*)T_i(t), & \text{ if } j = i \\ Q(V_a, w^*, h^*)S_j(t) + Q(V_g, w^*, h^*)(T_i(t) - S_j(t)), & \text{ if } j \neq i \end{cases} \quad (105)$$

The optimization problem is a min-max problem. $T_i(t)$ represents the time for i th UAV to clear the threat and return to the UGVs (with speed V_a). If the j th UAV is not assigned, $S_j(t)$ represents the time for it takes to return to the UGVs with speed

V_a . $T_i(t) - S_j(t)$ therefore represents the time for the j th UAV to fly alongside the UGVs with speed V_g until the assigned UAV flies back. If UAV j is flying alongside the UGV at time t , then $x_j(t) = x_g(t)$ and $S_j(t) = 0$. Hence, the quantity

$$f_j(t) + \begin{cases} Q(V_a, w^*, h^*)T_i(t), & \text{if } j = i \\ Q(V_a, w^*, h^*)S_j(t) + Q(V_g, w^*, h^*)(T_i(t) - S_j(t)), & \text{if } j \neq i \end{cases} \quad (106)$$

represents the accumulated fuel consumption of the j th UAV, assuming UAV i is assigned.

It is important to note here that the convoy protection problem formulation, along with the corresponding task assignment algorithm presented here, is independent of how the quantities $T_i(t)$ and $S_j(t)$ are computed for estimating the flight times of the UAVs. Nevertheless, for the sake of implementation, the hardware experiments presented later on will compute $T_i(t)$ and $S_j(t)$ assuming that the UAV flies in a straight line and that the UGV convoy's path can be approximated as a 2-D piecewise-linear path as defined by a sequence of waypoints. Interested readers may refer to [119] for the technical details.

As shown in Figure 27, $P_1(t)$ makes assignments such that the UAV with most accumulated fuel consumed after the clearance of the threat is minimized.

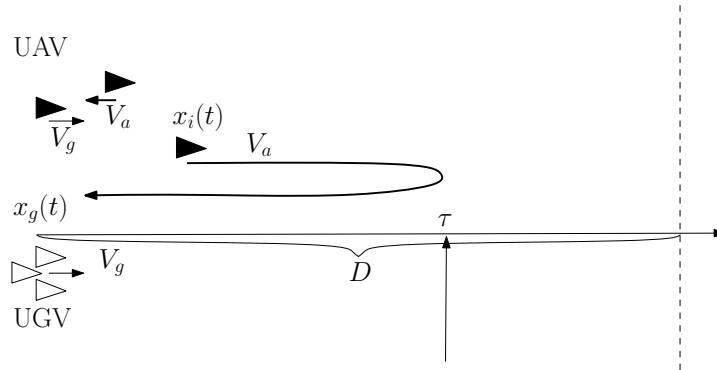


Figure 27: Selection Policy in action. In this example, the threat is not a pop-up

Let M denote the total number of UAVs (in this case $M = 3$). Min-max problems

are generally hard to solve, and the complexity of computing $P_1(t)$ is $O(M^2)$. However, under certain design choices of UAV speeds V_a and V_g , weight w^* , and altitude h^* , we can reduce the optimization problem to one that has $O(M)$ (linear) complexity. The main assumption needed for this is that it is more costly in terms of fuel consumption to be inspecting a threat than to remain with the convoy.

Theorem 6.2.2.1. *If UAV speeds V_a and V_g , payload weight w^* , and altitude h^* are chosen such that*

$$Q(V_a, w^*, h^*) > Q(V_g, w^*, h^*), \quad (107)$$

then the optimization problem $P_1(t)$ can be solved by another optimization problem:

$$P(t) = \arg \min_{i \in \{1,2,3\}} \{f_i(t) + Q(V_a, w^*, h^*)T_i(t)\}, \quad (108)$$

where $P(t) = P_1(t)$.

Proof. For the inner, maximum part of $P_1(t)$, denote its solution by $L(i, t)$ for a fixed i . In that case, $L(i, t) =$

$$\arg \max_{j \in \{1,2,3\}} f_j(t) + \begin{cases} Q(V_a, w^*, h^*)T_i(t), & \text{if } j = i \\ Q(V_a, w^*, h^*)S_j(t) + Q(V_g, w^*, h^*)(T_i(t) - S_j(t)), & \text{if } j \neq i \end{cases}. \quad (109)$$

Thus,

$$P_1(t) = \arg \min_{i \in \{1,2,3\}} \{L(i, t)\}. \quad (110)$$

If $L(i, t) = i, \forall i$, then:

$$P_1(t) = \arg \min_{i \in \{1,2,3\}} \{f_i(t) + Q(V_a, w^*, h^*)T_i(t)\} = P(t). \quad (111)$$

Note that since $Q(V_g, w^*, h^*) < Q(V_a, w^*, h^*)$,

$$Q(V_a, w^*, h^*)S_j(t) + Q(V_g, w^*, h^*)(T_i(t) - S_j(t)) < Q(V_a, w^*, h^*)T_i(t). \quad (112)$$

If $\exists i^*$ such that $L(i^*, t) \neq i^*$, this means that there is one UAV i^* such that, if it is assigned, the total fuel consumed by i^* is less than all other $j \neq i^*$ even though it consumed more fuel than all others during the time $T_{i^*}(t)$ because $Q(V_a, w^*, h^*)S_j(t) + Q(V_g, w^*, h^*)(T_i(t) - S_j(t)) < Q(V_a, w^*, h^*)T_i(t), \forall i, j$. Hence in this case

$$i^* = \arg \min_{i \in \{1,2,3\}} \{f_i(t) + Q(V_a, w^*, h^*)T_i(t)\} = \arg \min_{i \in \{1,2,3\}} \{L(i, t)\}, \quad (113)$$

and therefore:

$$P_1(t) = P(t). \quad (114)$$

□

To describe in words, $P(t)$ assigns the UAV so that, after the threat is cleared and returned to the UGVs, the total fuel consumed for the assigned UAV is minimum over all the UAVs. The projected time for the clearance of the threat is $T_i(t)$, where i denotes the assigned UAV. It should be noted that during this time, the UGVs has moved forward by $V_g T_i(t)$ distance.

6.2.2.4 Assignment Algorithm

Now we will present our task allocation algorithm which uses the above selection policy.

Algorithm: Task Allocation Algorithm

Initialize: Set t_0 .

Iterate: until all threats are cleared

If an event is triggered:

1. update $\bar{\tau}(t) = [\tau_1, \tau_2, \dots, \tau_{N(t)}]$.
2. find $\tau = \min_{j \in \{1, 2, \dots, N(t)\}} \tau_j$.
3. Solve $i^* = P(t)$ and use the solution i^* as the assignment to clear τ .
4. wait until another event is triggered.

Result: a task allocation algorithm that clears all threats (if none are hostile) and solves problem 1.

6.2.3 Experimental Validation

To showcase the robustness of the presented algorithm, as well as to demonstrate that it can indeed be successfully deployed in an actual environment characterized by numerous computational and communications limitations, we will implement the algorithm on a hardware testbed involving both UAVs and UGVs. In particular, the UAVs will consist of two Parrot AR.Drone quadrotors and the UGVs will be a team of Khepera III mobile robots. As shown in Figure 28, inter-vehicle messages and control signals are sent to the unmanned vehicles through a WiFi router. Moreover, the location and pose of each vehicle is tracked in real-time using a Vicon motion capture system.

The hardware demonstration to validate the algorithms developed here consists of a mission that is structured as follows:

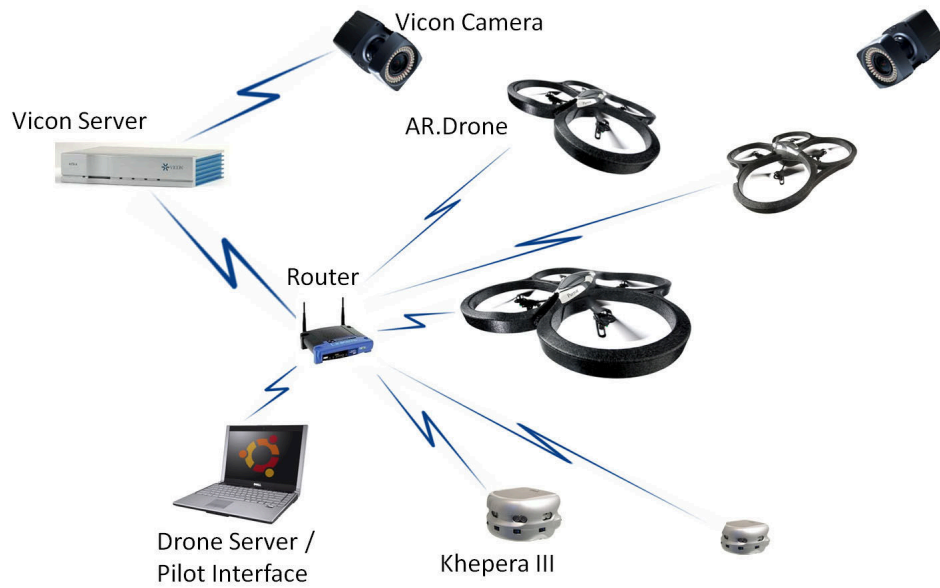


Figure 28: System-level architecture: Team of UAVs and UGVs are connected through a WiFi router and tracked using a Vicon motion capture system.

1. The UGVs are given a set of waypoints which define a path to be taken through the (possibly hostile) environment. All UAVs are assigned to protect the convoy by hovering over the convoy while maintaining a safe distance from one another.
2. Once a possible target has been identified. A single UAV is assigned to fly over the threat, while the remaining UAVs remain over the convoy.
3. After the threat is neutralized (cleared), the assigned UAV returns to once again protect the convoy.
4. In the event that a threat cannot be neutralized (is persistent), the UAV signals the convoy to replan its path to avoid the threat.

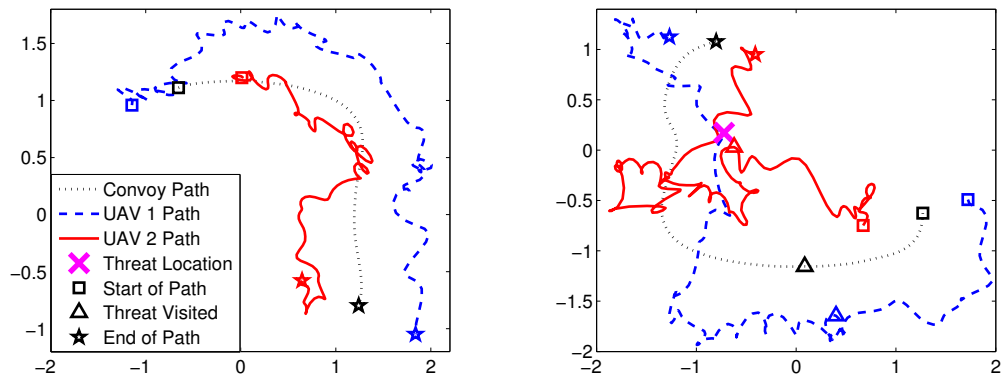
The plots in Figure 29 show the recorded trajectories of the UAVs and convoy during the experiment. In particular, Figure 29(a) shows both UAVs performing convoy protection by flying above the UGVs as they follow their intended path. Notice that the UAVs maintain a safe separation from one another. A photograph of

this operation is shown in Figure 30(a). Figure 29(b) shows the scenario when a non-persistent threat is encountered. In response, the UAVs execute the presented task allocation algorithm to determine which UAV gets dispatched to clear the threat so as to balance fuel consumption. Upon clearing the threat, the UAV returns to follow the convoy and the UGVs proceed in their originally intended path. A photograph showing a UAV examining the threat is shown in Figure 30(b).

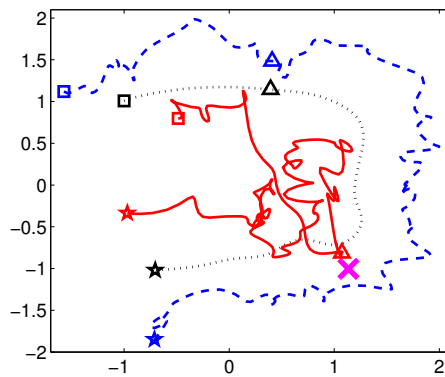
Figure 29(c) shows the convoy encountering a persistent threat which blocks its intended path. Once again, the UAVs execute the task allocation algorithm and dispatches a UAV to visit the threat. When it is determined that the threat is persistent, the UAV informs the convoy that it must replan the path so as to avoid the threat. Afterwards, the UAV returns to the convoy and the UGVs proceed in following the recomputed path. To further illustrate the replanning of the path, Figure 31(a) shows the originally intended path of the convoy corresponding to the trajectories in Figure 29(a) and 29(b). Figure 31(b) shows the recomputed path taken by the convoy so as to avoid the persistent threat.

To validate that the task allocation algorithm performs as expected, a separate experiment was conducted in which 16 threats were presented to the convoy over an extended period of time. Figure 32 shows the fuel consumption of the two UAVs over this time and marks when a UAV is dispatched to clear a threat. As seen in the plot, the algorithm successfully determines which UAV to dispatch when a threat is encountered such that the fuel consumption is balanced amongst the two UAVs over time.

This section presented a multi-UAV convoy protection solution for dynamic and hostile environments, where UAVs had to be dispatched to visit and clear pop-up threats. Since all UAVs could clear threats, the assignment of UAVs closely followed the concept presented in Chapter 3 on the controllability of homogeneous multi-agent systems. The UAVs had to be assigned to a permutation of the target point, which in

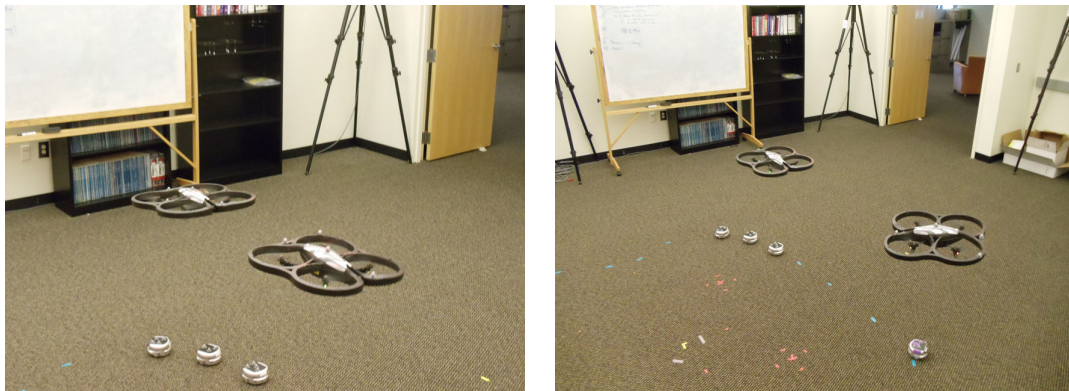


(a) Two UAVs perform convoy protection on (b) A UAV is dispatched to clear a non-persistent threat.



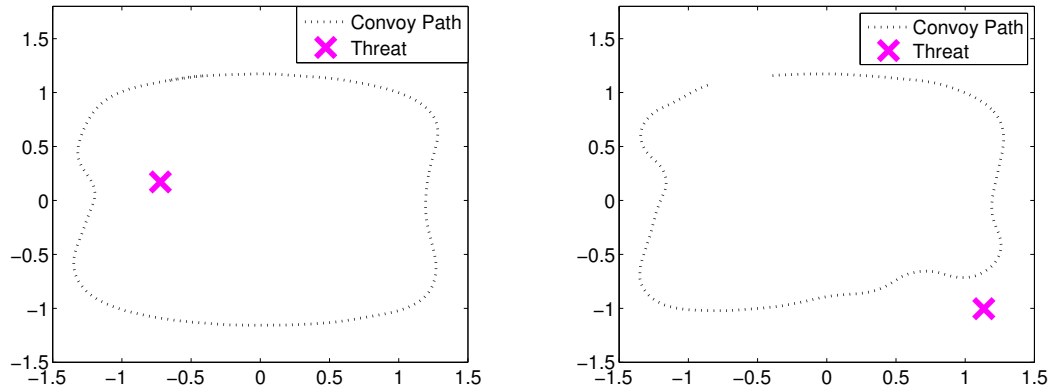
(c) A UAV identifies a persistent threat and the convoy replans its path.

Figure 29: Plots showing the trajectories of UAVs and the convoy which it protects as both a non-persistent and persistent threat are encountered.



(a) Two UAVs protect the convoy while maintaining spacing from one another. (b) One UAV is dispatched to visit a threat, while the other remains with the convoy.

Figure 30: Photos showing the convoy protection and threat neutralization as carried out by the hardware platform.



(a) The original intended path taken by the convoy, corresponding to Figures 29(a) and 29(b). (b) The recomputed path of the convoy upon being informed of a persistent threat, corresponding to Figure 29(c).

Figure 31: Plots showing the difference in path taken by the ground convoy before and after being informed about the presence of a persistent threat.

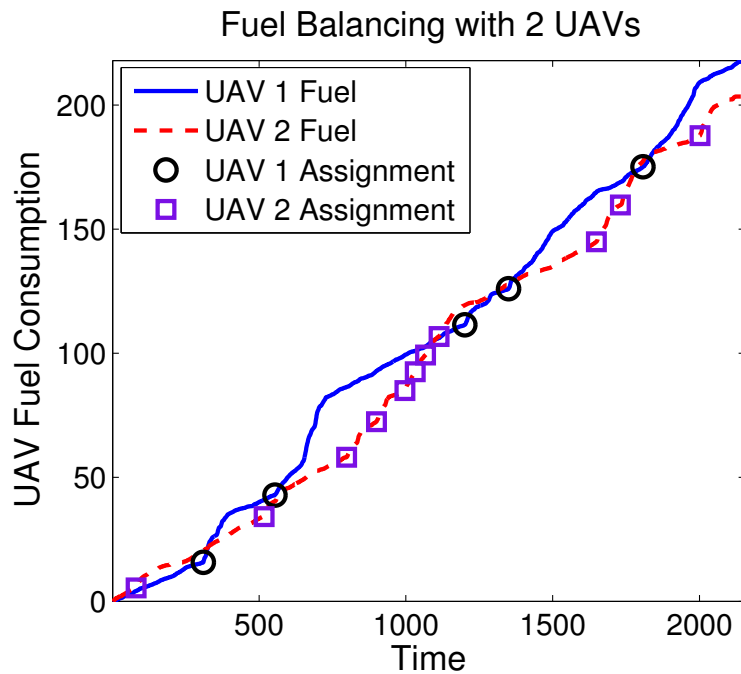


Figure 32: Plot of fuel consumption for the two UAVs illustrating that fuel usage is balanced while performing the convoy protection mission.

this case, consisted of positions above the threat and convoy. However, for practical purposes, the goal was to perform this assignment in a distributed manner such that the fuel consumption amongst UAVs was balanced, thereby maximizing the time between having to refuel. Finally, the solution was implemented and showcased using a hardware experiment involving Khepera III mobile robots and Parrot AR.Drone quadrotors.

6.3 Educational Tools for Robotics

The third and last application to be presented in this section is a demonstration of the principles behind the design methodology in Figure 2 to developing educational tools for robotics. As mentioned in Section 1.2.7 of the literature review, theory-based courses in systems and controls oftentimes have a hard time giving students an opportunity to use their skills in a more practical and applied setting. With the high-costs and maintenance associated with labs, many courses have turned to simulated virtual environments as a suitable alternative for experimentation. The educational tool to be presented here is a simulated multi-robot search-and-rescue mission within a virtual environment that has been used, as documented in [122], as the final project for the graduate-level ECE8823 networked controls course in both the Fall 2010 and Fall 2011 semesters at the Georgia Institute of Technology. The project required students to complete a complex mission by breaking it down into multiple subtasks and designing both decentralized controllers for performing each subtask, and switching strategies for the controllers. Using the same underlying concepts as found in the GPS Framework from Chapter 5, students then had to modify their decentralized controller and guard condition choices accordingly so that the entire sequence of controllers could be executed consecutively to complete the subtasks in order.

6.3.1 Tasks

In the last month of the semester, students in ECE8823 at the Georgia Institute of Technology received the following final project instructions:

The year is 2030 and NASA has identified an asteroid that is on a collision course with Earth! In order to deflect the asteroid, the scientists require samples from its surface to determine its physical composition. They have asked the robotics faculty at the Georgia Institute of Technology to plan a multi-robot expedition to collect samples from the asteroid's surface and bring them back to Earth for analysis. The robots managed to land on the asteroid successfully and were able to gather the samples. However, an unexpected pulse of electromagnetic radiation temporarily disabled the electronics on-board the robots, stranding them on the asteroid.

Based on your experience in networked controls from having taken ECE8823, members of the robotics faculty at the Georgia Institute of Technology have selected you to lead a rescue mission. Using the beacons placed on the surface of the asteroid from the first expedition for navigation, your mission is to design decentralized controllers for the multi-robot rescue team so as to:

- 1. Navigate a team of 6 robots through the rough terrain of the asteroid*
- 2. Locate and re-activate the 6 disabled robots from the first expedition*
- 3. Bring both robot teams back to the platform (leave no robot behind) and get into a specific formation to wait to be picked up by an orbiting spacecraft.*

Along with the instructions, students in ECE8823 were given MATLAB code to simulate a team of robots navigating through a virtual environment with 6 waypoints that must be cleared in order, as shown in Figure 33. For simplicity, the terrain is assumed to be flat and so each robot i has position $x_i \in \mathbb{R}^2$. Furthermore, the robots are assumed to have single integrator dynamics with actuator saturation, and



Figure 33: The virtual environment used for the multi-robot search and rescue final project. Students must design decentralized controllers to navigate a team of robots through all 6 waypoints in order, where each waypoint challenges the student to apply a different concept learned throughout the class.

can only sense and update their controllers every T seconds. Each of the robots are equipped with omnidirectional sensors, allowing them to detect neighboring robots and obstacles that are within a distance Δ . The network topology at time t can therefore be represented by an undirected graph $G(t) = (V(t), E(t))$ where $V(t) = \{1, \dots, N\}$ is the vertex set, with each vertex corresponding to the agent in the network with the same index. The edge set $E(t) \subset V(t) \times V(t)$ is such that an unordered pair $(i, j) \in E(t)$ if and only if $\|x_i(t) - x_j(t)\| \leq \Delta$.

For each of the waypoints, students were tasked to write a single decentralized controller that would run on all the robots simultaneously and drive the multi-robot team so as to clear that waypoint. MATLAB function templates were provided to the

students for writing the 6 decentralized controllers used to clear the 6 waypoints. To ensure the coordination algorithms that students designed were indeed decentralized, each function must calculate a robot’s control signal while taking as input the robot’s unique ID, state, list of neighboring robots as given by the current network topology, relative displacement measurements within the robot’s local coordinate frame to each neighbor and to nearby obstacles, a flag indicating whether this was the first time the robot has executed the controller, and locally stored information within each robot’s limited memory.

Since robots have a physical radius D , each controller must make use of the locally available information to drive the robots so as to avoid any collisions with both the environment and neighboring robots. Furthermore, to help the robots navigate between the waypoints, a single “leader” robot is also given the ability to sense its relative displacement to the most current waypoint and use that information when computing its control. A waypoint is cleared when all previous waypoints have been cleared, a robot is at the current waypoint, and the current network topology is connected, i.e., a path of edges exists between each pair of robots. Each waypoint requires the robots to perform a different maneuver to complete, and thus challenges students to apply different concepts learned throughout the class. Moreover, additional locally-checkable guard conditions could be implemented by students to specify when agents would switch from one controller to another.

6.3.1.1 Subtasks

Referring to Figure 34(a), the first waypoint requires the robots to simply move from one point to another without colliding with each other or the environment. Looking at the map in Figure 33, the robots then must travel through a narrow valley to reach Waypoint 2, and then navigate through a field littered with small obstacles to reach Waypoint 3. To clear Waypoint 4, the robots must search a bounded area to recover

the 6 stranded robots from the previous mission. All 12 robots must then perform a “splitting and merging” maneuver around a large obstacle to reach Waypoint 5. Finally, the robots must all move onto the platform on Waypoint 6 and get into a particular formation, as shown in Figure 34(i), to await rescue.

6.3.1.2 Decentralized Control Strategy

Notice how clearing each waypoint ultimately requires that each robot in the network avoid obstacles and other robots, while maintaining network connectivity, and move towards the next waypoint. Waypoint 1 was designed to give the students a chance to solidify their solution to this problem before moving on to more difficult tasks. The topic of weighted consensus protocols was discussed in class, where each robot i moves with a velocity vector that is a weighted combination of the relative displacement vectors between each of its neighbors:

$$\dot{x}_i(t) = - \sum_{j \in N_i(t)} w_{ij}(t) (x_i(t) - x_j(t)), \quad (115)$$

where $N_i(t) = \{j \mid (i, j) \in E(t)\}$ is the set of robot i 's current neighbors. By choosing the weight function $w_{ij}(t)$ carefully, it was shown in class, using Lyapunov-based arguments, that robots can be made to preserve network connectivity and maintain fixed distances from one another. Most students in the class had chosen to combine the weights for network connectivity preservation with additional weights they constructed to repel neighboring robots away from each other if they got too close. Students were then able to make robots avoid nearby obstacles by treating them as virtual agents, from which they needed to maintain a certain distance from. The leader robot could balance the objective of moving towards the next waypoint with the secondary objective of avoiding other obstacles and robots, by having the next waypoint act as a stationary virtual neighbor. Using these and similar methods, students were able to successfully clear Waypoint 1, as shown in Figure 34(b).

To clear waypoint 2, most students realized that the valley was too narrow to

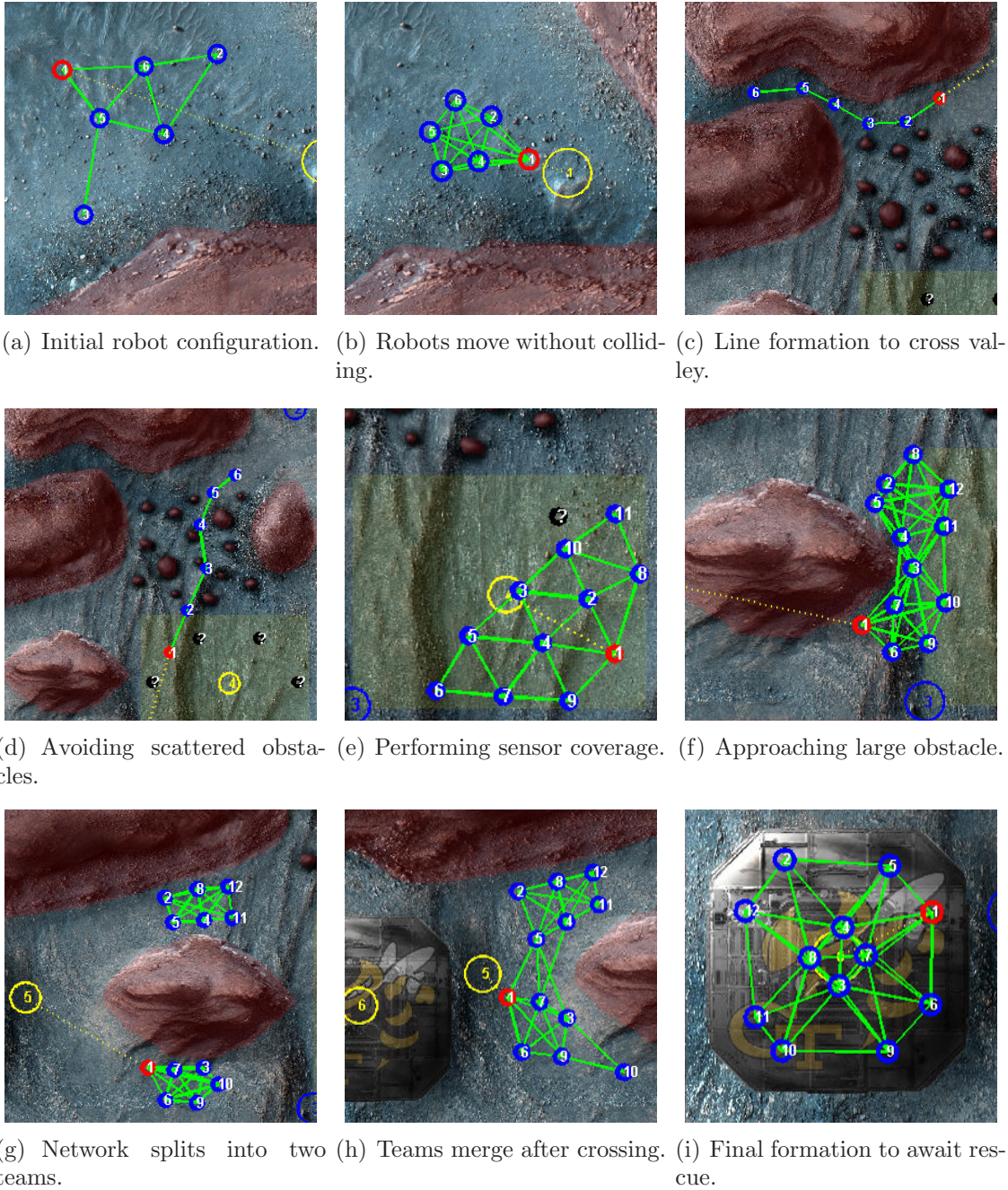


Figure 34: Screenshots showing the students' solutions for solving all 6 waypoints in the multi-robot search and rescue mission.

fit all the robots through at once. Methods for network topology control [72], as discussed in class, could be used to make the robots squeeze through the valley in a line configuration as shown in Figure 34(c). Clearing waypoint 3 also required obstacle avoidance, but students had to solve the problem of maintaining network connectivity even if different robots chose to take different paths through the field as seen in Figure 34(d). Figure 34(e) shows students clearing waypoint 4 by using Voronoi-based sensor coverage algorithms [26] to search the enclosed area for the stranded robots. Since a stranded robot reactivates and joins the network if another robot gets close enough to it, the sensor coverage algorithms used had to be scalable as robots dynamically join the network.

Waypoint 5 was the most difficult for students to clear since the “splitting and merging” maneuver, shown in Figures 34(f), 34(g), and 34(h), caused the network to become disconnected as robots go around opposite sides of the obstacle. Most of the solutions that the students came up with fell into one of two categories. The first approach was to have the leader robot move towards the next waypoint and for all other robots to maintain a sense of “momentum”. Therefore, even if the network becomes disconnected, robots would still have an idea of the overall direction that they should be moving in. Another approach was to have the leader robot move towards the next waypoint at a speed proportional to its distance from it. The neighboring robots could then estimate the position of the next waypoint through the leader’s actions and become virtual leaders themselves. Finally, to clear waypoint 6, students implemented many different heuristic-based methods to solve the distributed assignment problem of moving a robot to each target point, without the use of any inter-robot communication.

6.3.1.3 Executing the Sequence of Decentralized Controllers

The virtual environment allowed students to save the system state upon clearing a waypoint so that students could, for example, work on the controller for clearing waypoint 4 without having the simulation run through clearing waypoints 1 through 3 every time. This feature allowed students to concentrate on the design and testing of a single subtask-specific decentralized controller at a time. Oftentimes, students may even go back and refine the controllers used to clear earlier waypoints in order to get better performance. However, many students were in for a surprise when they tried to execute the entire sequence of decentralized controllers back-to-back.

For the same reason why arbitrary controllers could not be strung together to create an executable sequence in the GPS framework of Chapter 5, the decentralized controllers used to perform each subtask in this project would not necessarily work when executed consecutively. This is because many controllers would only achieve their intended results on the system if a specific initial network topology was in place. However, this required network topology may not necessarily have been provided at the termination of the previous controller.

To fix this problem, students had a number of options. One way was to make their decentralized controllers more robust to a wider range of initial network topologies. This can be seen as expanding the initial graph set \mathbb{G} of the decentralized consistent atom associated with each controller in the GPS framework. Another way was to change the guard condition for terminating the controllers so that they ended with a network topology that was desirable for the next controller in the sequence. Such a strategy can be seen as changing the final graph set \mathbb{H} and locally-checkable termination condition \mathcal{C} in the associated decentralized consistent atom. Finally, if all else failed, the students could try using a completely different controller to get the subtask done, which corresponded to removing an atom from the scripted GPS mode

sequence and inserting in a new one from the atom library. Therefore, without realizing it, the students were designing their decentralized control strategies for this project based off of the methodology in Figure 2.

6.3.2 Educational Outcomes

Of the 39 students in the Fall 2010 semester of ECE8823, 27 were able to complete all six waypoints successfully. Moreover, of the 19 students in the Fall 2011 semester of ECE8823, 18 were able to complete all the waypoints. During the last day of class for each semester, a half hour was dedicated to having students share and discuss their solutions with the entire class. The five fastest solutions in the class were then unveiled one at a time, as students cheered and applauded for their fellow classmates. Finally, an optional survey was given to the class for some feedback on the project and the responses were overwhelmingly positive. 30 of the 39 students in the Fall 2010 semester and 14 of the 19 students in the Fall 2011 semester responded. Out of these survey results, all students stated that the project helped solidify the concepts which were learned in class, and that they recommend the project be continued again next year. On average, students reported spending around 4 days (32 work hours) to complete the project. When asked what they had learned from the experience, the students commented on learning “how to combine different concepts in class for a working system,” and felt that the project allowed them to “understand the ideas/subjects learned in the lecture more rigidly.” Since the search and rescue mission differed from the simpler numerical simulations done in previous homeworks in that it required the robots to balance performing a large number of tasks simultaneously, students also reported learning “how to actually design for real situations, and how difficult it is to balance gains.”

To summarize, this chapter presented three applications of the design methodology in Figure 2, which is supported by the tool suite that was developed throughout this

dissertation. In particular, the underlying ideas behind the tools in this dissertation have been applied to solve a series of realistic and complex problems. First, a distributed aircraft merging and spacing solution was developed for terminal approaches, which allowed for conflicting aircraft to resolve conflicts using dual decomposition and come to an agreement on safe merging times and trajectories under the framework of the FAA's NextGen. Second, a collaborative multi-UAV convoy protection problem was solved and showcased using Khepera III UGVs and Parrot AR.Drone UAVs that ran a distributed fuel-balancing task assignment algorithm in order to eliminate the danger of hazardous pop-up threats. Finally, a project for a graduate-level course on networked controls was presented where students were tasked with designing decentralized control laws and switching strategies for completing a simulated multi-agent search-and-rescue mission.

CHAPTER VII

CONCLUSION

This dissertation presented a suite of tools for the design of multi-agent systems in the following three application areas: network design, generation of decentralized controllers, and the high-level scripting of decentralized controller sequences. On the topic of network design, agent heterogeneity and its influence on the expressiveness of the multi-agent system as a whole was investigated. First, a precise definition of heterogeneity was presented and from it, a metric was developed to quantify heterogeneity specifically for multi-agent systems. Then, the controllability properties of homogeneous single-leader networks were analyzed in depth to understand how heterogeneity (or lack thereof) influenced the expressiveness of a system. Next, an optimal decentralization algorithm was presented for the task of generating decentralized controllers. Given a set of parameterized decentralized control laws, the algorithm used optimal control techniques to find the parameters that allowed for agents to track a desired multi-agent motion the best. The resulting algorithm was showcased in a simulation where agents had to imitate a complex drumline-inspired multi-agent dance. Finally, the Graph Process Specification (GPS) framework was presented as a way for designers to script, at a high-level, sequences of decentralized controllers for agents to execute consecutively. Using the concept of an atom, decentralized controllers could be sequenced so as to ensure that network topological requirements for each of the controllers were satisfied throughout execution.

The developed tools were shown to support a common multi-agent design methodology that was showcased in three applications. First, a distributed aircraft merging and spacing solution was presented where conflicting aircraft negotiated with each

other using the FAA's NextGen ADS-B protocol in order to agree on flight plans which ensured safety. The resulting solution was tested in a simulated environment involving the merging of three legs of flight into one, where aircraft had heterogeneous spacing requirements. Second, a collaborative multi-UAV convoy protection problem in a dynamic and hostile environment was addressed, where UAVs had to monitor a UGV convoy and be dispatched to clear pop-up threats. The developed UAV assignment algorithm was tested in a hardware experiment involving a UGV convoy consisting of Khepera III mobile robots, and a team of Parrot AR.Drone quadrotor UAVs. The last presented application was an educational tool for the graduate-level ECE8823 course on networked controls at the Georgia Institute of Technology, where students had to implement a sequence of decentralized controllers to complete a series of subtasks in a simulated multi-robot search-and-rescue mission. Decentralized control strategies developed by students for the project were discussed, and feedback from surveyed data for both the Fall 2010 and Fall 2011 semesters was presented.

REFERENCES

- [1] AHMADZADEH, A., BUCHMAN, G., CHENG, P., JADBABAIE, A., KELLER, J., KUMAR, V., and PAPPAS, G., “Cooperative control of uavs for search and coverage,” in *Proceedings of the AUVSI conference on unmanned systems*, 2006.
- [2] ALAMIR, M. and ATTIA, S., “On solving optimal control problems for switched hybrid nonlinear systems by strong variations algorithms,” in *Proceedings of 6th IFAC symposium on nonlinear control systems*, pp. 558–563, 2004.
- [3] ALOISE, D., DESHPANDE, A., HANSEN, P., and POPAT, P., “NP-hardness of Euclidean sum-of-squares clustering,” *Machine Learning*, vol. 75, no. 2, pp. 245–248, 2009.
- [4] ANTSAKLIS, P., “A brief introduction to the theory and applications of hybrid systems,” in *Proc IEEE, Special Issue on Hybrid Systems: Theory and Applications*, Citeseer, 2000.
- [5] ARSIE, A., SAVLA, K., and FRAZZOLI, E., “Efficient routing algorithms for multiple vehicles with no explicit communications,” *Automatic Control, IEEE Transactions on*, vol. 54, no. 10, pp. 2302–2317, 2009.
- [6] ARSLAN, G., MARDEN, J., and SHAMMA, J., “Autonomous vehicle-target assignment: A game-theoretical formulation,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, p. 584, 2007.
- [7] ATTIA, S., ALAMIR, M., and DE WIT, C., “Sub optimal control of switched nonlinear systems under location and switching constraints,” in *IFAC World Congress*, 2005.
- [8] AVIATION COMMUNICATION & SURVEILLANCE SYSTEMS, “Saferoute ads-b program briefing.” http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/enroute/surveillance_broadcast/program_office_news/industry_day_8-28/media/ACSS.pdf, 2006. [Online; accessed 10-Jan.-2012].
- [9] AXELSSON, H., WARDI, Y., EGERSTEDT, M., and VERRIEST, E., “Gradient descent approach to optimal mode scheduling in hybrid dynamical systems,” *Journal of Optimization Theory and Applications*, vol. 136, no. 2, pp. 167–186, 2008.
- [10] BALAKIRSKY, S., CARPIN, S., KLEINER, A., LEWIS, M., VISSER, A., WANG, J., and ZIPARO, V., “Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from robocup rescue,” *Journal of Field Robotics*, vol. 24, no. 11-12, pp. 943–967, 2007.

- [11] BALCH, T., “Hierarchic social entropy: An information theoretic measure of robot group diversity,” *Autonomous robots*, vol. 8, no. 3, pp. 209–238, 2000.
- [12] BALCH, T., SUMMET, J., BLANK, D., KUMAR, D., GUZDIAL, M., O’HARA, K., WALKER, D., SWEAT, M., GUPTA, G., TANSLEY, S., and OTHERS, “Designing personal robots for education: Hardware, software, and curriculum,” *IEEE Pervasive Computing*, pp. 5–9, 2008.
- [13] BAMIEH, B., PAGANINI, F., and DAHLEH, M., “Distributed control of spatially invariant systems,” *Automatic Control, IEEE Transactions on*, vol. 47, no. 7, pp. 1091–1107, 2002.
- [14] BEARD, R., MCLAIN, T., GOODRICH, M., and ANDERSON, E., “Coordinated target assignment and intercept for unmanned air vehicles,” *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 6, pp. 911–922, 2002.
- [15] BEMPORAD, A., BORRELLI, F., and MORARI, M., “Piecewise linear optimal controllers for hybrid systems,” in *American Control Conference, 2000. Proceedings of the 2000*, vol. 2, pp. 1190–1194, IEEE, 2000.
- [16] BRADLEY, P., BENNETT, K., and DEMIRIZ, A., “Constrained k-means clustering,” in *MSR-TR-2000-65, Microsoft Research*, Citeseer, 2000.
- [17] BRANICKY, M., BORKAR, V., and MITTER, S., “A unified framework for hybrid control: Model and optimal control theory,” *Automatic Control, IEEE Transactions on*, vol. 43, no. 1, pp. 31–45, 1998.
- [18] BROCKETT, R., “On the computer control of movement,” in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pp. 534–540, IEEE, 1987.
- [19] BROUCKE, M., “Disjoint path algorithms for planar reconfiguration of identical vehicles,” in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3, pp. 2199–2204, IEEE, 2003.
- [20] BROUCKE, M., “Reconfiguration of identical vehicles in 3D,” in *Proceedings of the 42nd IEEE Conference on Decision and Control 2003*, vol. 2, pp. 1538–1543, 2003.
- [21] BURKARD, R., “Selected topics on assignment problems,” *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 257–302, 2002.
- [22] CARPIN, S., LEWIS, M., WANG, J., BALAKIRSKY, S., and SCRAPPER, C., “USARSim: a robot simulator for research and education,” in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1400–1405, IEEE, 2007.

- [23] CHIPALKATTY, R., TWU, P., RAHMANI, A., and EGERSTEDT, M., “Distributed scheduling for heterogeneous air traffic merging and spacing during the terminal phase of flight,” *AIAA Journal of Guidance, Control, and Dynamics*, *Accepted on Feb. 2012*.
- [24] CHIPALKATTY, R., TWU, P., RAHMANI, A., and EGERSTEDT, M., “Distributed scheduling for air traffic throughput maximization during the terminal phase of flight,” in *Decision and Control (CDC), 2010 49th IEEE Conference on*, pp. 1195–1200, IEEE, 2010.
- [25] CHOI, H., BRUNET, L., and HOW, J., “Consensus-based decentralized auctions for robust task allocation,” *Robotics, IEEE Transactions on*, vol. 25, no. 4, pp. 912–926, 2009.
- [26] CORTES, J., MARTINEZ, S., KARATAS, T., and BULLO, F., “Coverage control for mobile sensing networks,” *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 2, pp. 243–255, 2004.
- [27] COUZIN, I. and FRANKS, N., “Self-organized lane formation and optimized traffic flow in army ants,” *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 270, no. 1511, p. 139, 2003.
- [28] COUZIN, I., KRAUSE, J., FRANKS, N., and LEVIN, S., “Effective leadership and decision-making in animal groups on the move,” *Nature*, vol. 433, no. 7025, pp. 513–516, 2005.
- [29] COWELL, F., “Measurement of inequality,” *Handbook of income distribution*, vol. 1, pp. 87–166, 2000.
- [30] CUCKER, F. and SMALE, S., “Emergent behavior in flocks,” *Automatic Control, IEEE Transactions on*, vol. 52, no. 5, pp. 852–862, 2007.
- [31] DE SILVA, V. and GHRIST, R., “Coverage in sensor networks via persistent homology,” *Algebraic & Geometric Topology*, vol. 7, pp. 339–358, 2007.
- [32] DURFEE, W., LI, P., and WALETZKO, D., “At-home system and controls laboratories,” in *Proceedings of the American Society of Engineering Education Annual Conference & Exposition*, 2005.
- [33] EGERSTEDT, M. and HU, X., “Formation control with virtual leaders and reduced communications,” *IEEE Trans. on Robotics and Automation*, vol. 17, no. 6, pp. 947–951, 2001.
- [34] EGERSTEDT, M., WARDI, Y., and AXELSSON, H., “Transition-time optimization for switched-mode dynamical systems,” *Automatic Control, IEEE Transactions on*, vol. 51, no. 1, pp. 110–115, 2006.

- [35] EREN, T., WHITELEY, W., ANDERSON, B., MORSE, A., and BELHUMEUR, P., “Information structures to secure control of rigid formations with leader-follower architecture,” in *American Control Conference, 2005. Proceedings of the 2005*, pp. 2966–2971, IEEE, 2005.
- [36] ERZBERGER, H., “Transforming the nas: The next generation air traffic control system,” in *Proceedings of the 24th Int. Congress of the Aeronautical Sciences (ICAS)*, 2004.
- [37] FAVENNEC, B., HOFFMAN, E., TRZMIEL, A., VERGNE, F., and ZEGHAL, K., “The point merge arrival flow integration technique: Towards more complex environments and advanced continuous descent,” in *9th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, 2009.
- [38] FAX, J. and MURRAY, R., “Information flow and cooperative control of vehicle formations,” *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1465–1476, 2004.
- [39] FERRI, B., AHMED, S., MICHAELS, J., DEAN, E., GARYET, C., and SHEARMAN, S., “Signal processing experiments with the LEGO MINDSTORMS NXT kit for use in signals and systems courses,” in *American Control Conference, 2009. ACC’09.*, pp. 3787–3792, IEEE, 2009.
- [40] GODSIL, C. and ROYLE, G., *Algebraic Graph Theory*. Springer Verlag, 2001.
- [41] GROCHOLSKY, B., KELLER, J., KUMAR, V., and PAPPAS, G., “Cooperative air and ground surveillance,” *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 16–25, 2006.
- [42] HECK, B., CLEMENTS, N., and FERRI, A., “A LEGO experiment for embedded control system design,” *Control Systems Magazine, IEEE*, vol. 24, no. 5, pp. 61–64, 2004.
- [43] HEDLUND, S. and RANTZER, A., “Optimal control of hybrid systems,” in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 4, pp. 3972–3977, IEEE, 1999.
- [44] JADBABAIE, A., LIN, J., and MORSE, A., “Coordination of groups of mobile autonomous agents using nearest neighbor rules,” *Automatic Control, IEEE Transactions on*, vol. 48, no. 6, pp. 988–1001, 2003.
- [45] JAMESON, S., FRANKE, J., SZCZERBA, R., and STOCKDALE, S., “Collaborative autonomy for manned/unmanned teams,” in *Annual Forum Proceedings-American Helicopter Society*, vol. 61, p. 1673, American Helicopter Society, Inc, 2005.

- [46] JAMESON, S. and STONEKING, C., “Army aviation situational awareness through intelligent agent-based discovery, propagation, and fusion of information,” in *Annual Forum Proceedings-American Helicopter Society*, vol. 58, pp. 1350–1360, American Helicopter Society, Inc, 2002.
- [47] JI, M., AZUMA, S., and EGERSTEDT, M., “Role-assignment in multi-agent coordination,” *International Journal of Assistive Robotics and Mechatronics*, vol. 7, no. 1, pp. 32–40, 2006.
- [48] JI, M. and EGERSTEDT, M., “Distributed coordination control of multi-agent systems while preserving connectedness,” *Robotics, IEEE Transactions on*, vol. 23, no. 4, pp. 693–703, 2007.
- [49] JOHANSSON, K., EGERSTEDT, M., LYGEROS, J., and SASTRY, S., “On the regularization of zeno hybrid automata,” *Systems & Control Letters*, vol. 38, no. 3, pp. 141–150, 1999.
- [50] JOHNSON, D. and GAREY, M., “Computers and intractability: A guide to the theory of np-completeness,” *Freeman&Co, San Francisco*, 1979.
- [51] JUSTH, E. and KRISHNAPRASAD, P., “Equilibria and steering laws for planar formations* 1,” *Systems & Control Letters*, vol. 52, no. 1, pp. 25–38, 2004.
- [52] KIENER, J. and VON STRYK, O., “Towards cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots,” *Robotics and Autonomous Systems*, vol. 58, no. 7, pp. 921–929, 2010.
- [53] KITTS, C. and EGERSTEDT, M., “Design, control, and applications of real-world multirobot systems [from the guest editors],” *Robotics Automation Magazine, IEEE*, vol. 15, no. 1, p. 8, 2008.
- [54] KLOETZER, M. and BELTA, C., “Temporal logic planning and control of robotic swarms by hierarchical abstractions,” *Robotics, IEEE Transactions on*, vol. 23, no. 2, pp. 320–330, 2007.
- [55] KOUTSOUKOS, X., ANTSAKLIS, P., STIVER, J., and LEMMON, M., “Supervisory control of hybrid systems,” *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1026–1049, 2000.
- [56] KUHN, H., “The hungarian method for the assignment problem,” *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.
- [57] LAWTON, J., BEARD, R., and YOUNG, B., “A decentralized approach to formation maneuvers,” *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 6, pp. 933–941, 2003.
- [58] LIN, Z., BROUCKE, M., and FRANCIS, B., “Local control strategies for groups of mobile autonomous agents,” *Automatic Control, IEEE Transactions on*, vol. 49, no. 4, pp. 622–629, 2004.

- [59] LIN, Z., FRANCIS, B., and MAGGIORE, M., “Necessary and sufficient graphical conditions for formation control of unicycles,” *Automatic Control, IEEE Transactions on*, vol. 50, no. 1, pp. 121–127, 2005.
- [60] LIU, X., LIN, H., and CHEN, B., “A graph-theoretic characterization of structural controllability for multi-agent system with switching topology,” in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 7012–7017, IEEE, 2009.
- [61] LIU, Y., PASSINO, K., and POLYCARPOU, M., “Stability analysis of one-dimensional asynchronous swarms,” *Automatic Control, IEEE Transactions on*, vol. 48, no. 10, pp. 1848–1854, 2003.
- [62] LLOYD, S., “Least squares quantization in pcm,” *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.
- [63] LYNCH, N., *Distributed algorithms*. Morgan Kaufmann, 1996.
- [64] MANIKONDA, V., KRISHNAPRASAD, P., and HENDLER, J., “Languages, behaviors, hybrid architectures and motion control,” *Mathematical control theory*, pp. 199–226, 1998.
- [65] MAO, Z., DUGAIL, D., FERON, E., and BILIMORIA, K., “Stability of intersecting aircraft flows using heading-change maneuvers for conflict avoidance,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 6, no. 4, pp. 357–369, 2005.
- [66] MARTIN, P., DE LA CROIX, J., and EGERSTEDT, M., “MDLn: A motion description language for networked systems,” in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 558–563, IEEE, 2008.
- [67] MARTINEZ, S., CORTES, J., and BULLO, F., “Motion coordination with distributed information,” *Control Systems Magazine, IEEE*, vol. 27, no. 4, pp. 75–88, 2007.
- [68] MARTINI, S., EGERSTEDT, M., and BICCHI, A., “Controllability analysis of multi-agent systems using relaxed equitable partitions,” *International Journal of Systems, Control and Communications*, vol. 2, no. 1, pp. 100–121, 2010.
- [69] McNALLY, D. and GONG, C., “Concept and laboratory analysis of trajectory-based automation for separation assurance,” *Air Traffic Control Quarterly*, vol. 15, no. 1, pp. 35–63, 2007.
- [70] McNEW, J. and KLAVINS, E., “Locally interacting hybrid systems with embedded graph grammars,” in *Decision and Control, 2006 45th IEEE Conference on*, pp. 6080–6087, IEEE, 2006.

- [71] MCNEW, J., KLAVINS, E., and EGERSTEDT, M., “Solving coverage problems with embedded graph grammars,” *Hybrid Systems: Computation and Control*, pp. 413–427, 2007.
- [72] MESBAHI, M. and EGERSTEDT, M., *Graph theoretic methods in multiagent networks*. Princeton Univ Pr, 2010.
- [73] MICHEL, O., “Cyberbotics Ltd. Webots™: Professional mobile robot simulation,” *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.
- [74] MOORE, B. and PASSINO, K., “Distributed balancing of aavs for uniform surveillance coverage,” in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC’05. 44th IEEE Conference on*, pp. 7060–7065, IEEE, 2005.
- [75] MORSE, A., *Control using logic-based switching*. Citeseer, 1997.
- [76] MOTEE, N. and JADBABAIE, A., “Optimal control of spatially distributed systems,” *Automatic Control, IEEE Transactions on*, vol. 53, no. 7, pp. 1616–1629, 2008.
- [77] MUHAMMAD, A. and EGERSTEDT, M., “Connectivity graphs as models of local interactions,” *Applied mathematics and computation*, vol. 168, no. 1, pp. 243–269, 2005.
- [78] MUHAMMAD, A. and JADBABAIE, A., “Dynamic coverage verification in mobile sensor networks via switched higher order Laplacians,” in *Robotics: Science & Systems*, Citeseer, 2007.
- [79] MURPHEY, T., “Teaching rigid body mechanics using student-created virtual environments,” *Education, IEEE Transactions on*, vol. 51, no. 1, pp. 45–52, 2008.
- [80] NEXTGEN INTEGRATION AND IMPLEMENTATION OFFICE, FAA, “Nextgen implementation plan,” tech. rep., Washington, DC: FAA, 2010.
- [81] NORTHROP GRUMMAN, “How does hart work?.” http://www.as.northropgrumman.com/products/hart/assets/brochure_HART.pdf, 2011. [Online; accessed 11-Jan.-2012].
- [82] OLFATI-SABER, R., “Flocking for multi-agent dynamic systems: Algorithms and theory,” *Automatic Control, IEEE Transactions on*, vol. 51, no. 3, pp. 401–420, 2006.
- [83] OLFATI-SABER, R., FAX, J., and MURRAY, R., “Consensus and cooperation in networked multi-agent systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

- [84] OLFATI-SABER, R. and MURRAY, R., “Consensus problems in networks of agents with switching topology and time-delays,” *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1520–1533, 2004.
- [85] PAGELS, M. A., “Heterogeneous airborne reconnaissance team (hart).” [http://www.docstoc.com/docs/52321907/Heterogeneous-Airborne-Reconnaissance-Team-\(HART\)-August-2008](http://www.docstoc.com/docs/52321907/Heterogeneous-Airborne-Reconnaissance-Team-(HART)-August-2008), 2008. [Online; accessed 11-Jan.-2012].
- [86] PALOMAR, D. and CHIANG, M., “A tutorial on decomposition methods for network utility maximization,” *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 8, pp. 1439–1451, 2006.
- [87] PAPADIMITRIOU, C., *Computational complexity*. John Wiley and Sons Ltd., 2003.
- [88] PARDALOS, P., RENDL, F., and WOLKOWICZ, H., “The quadratic assignment problem: A survey and recent developments,” in *In Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, volume 16 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1994.
- [89] PAVOINE, S., OLLIER, S., and PONTIER, D., “Measuring diversity from dissimilarities with Rao’s quadratic entropy: Are any dissimilarities suitable?,” *Theoretical Population Biology*, vol. 67, no. 4, pp. 231–239, 2005.
- [90] PAVONE, M., FRAZZOLI, E., and BULLO, F., “Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment,” *Automatic Control, IEEE Transactions on*, no. 99, pp. 1–1, 2009.
- [91] PIMENTA, L., KUMAR, V., MESQUITA, R., and PEREIRA, G., “Sensing and coverage for a network of heterogeneous robots,” in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 3947–3952, IEEE, 2008.
- [92] PONDA, S., REDDING, J., CHOI, H., HOW, J., VAVRINA, M., and VIAN, J., “Decentralized planning for complex missions with dynamic communication constraints,” in *American Control Conference (ACC), 2010*, pp. 3998–4003, IEEE, 2010.
- [93] PSARAFTIS, H., “Dynamic vehicle routing problems,” *Vehicle routing: Methods and studies*, vol. 16, pp. 223–248, 1988.
- [94] RAHMANI, A., JI, M., MESBAHI, M., and EGERSTEDT, M., “Controllability of multi-agent systems from a graph-theoretic perspective,” *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 162–186, 2009.
- [95] RAHMANI, A., KOSUGE, K., TSUKAMAKI, T., and MESBAHI, M., “Multiple uav deconfliction via navigation functions,” in *AIAA Guidance, Navigation and Control Conference*, 2008.

- [96] RANTZER, A., “A separation principle for distributed control,” in *Decision and Control, 2006 45th IEEE Conference on*, pp. 3609–3613, IEEE, 2007.
- [97] RANTZER, A., “Dynamic dual decomposition for distributed control,” in *American Control Conference, 2009. ACC’09.*, pp. 884–888, IEEE, 2009.
- [98] RAO, C., “Diversity and dissimilarity coefficients: A unified approach* 1,” *Theoretical Population Biology*, vol. 21, no. 1, pp. 24–43, 1982.
- [99] REN, W. and BEARD, R., “Consensus seeking in multiagent systems under dynamically changing interaction topologies,” *Automatic Control, IEEE Transactions on*, vol. 50, no. 5, pp. 655–661, 2005.
- [100] REYNOLDS, C., “Flocks, herds and schools: A distributed behavioral model,” in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 25–34, ACM, 1987.
- [101] RICHARDS, A., BELLINGHAM, J., TILLERSON, M., and HOW, J., “Coordination and control of multiple uavs,” in *AIAA guidance, navigation, and control conference, Monterey, CA*, 2002.
- [102] ROTKOWITZ, M. and LALL, S., “A characterization of convex problems in decentralized Control,” *Automatic Control, IEEE Transactions on*, vol. 51, no. 2, pp. 274–286, 2006.
- [103] ROY, K. and TOMLIN, C., “Enroute airspace control and controller workload analysis using a novel slot-based sector model,” in *American Control Conference, 2006*, pp. 6–pp, IEEE, 2006.
- [104] RTCA, INC., “Minimum operational performance standards (mops) for aircraft surveillance applications system (asas),” Tech. Rep. DO-317, Washington, DC, 2009.
- [105] SCHUMACHER, C., CHANDLER, P., RASMUSSEN, S., and WALKER, D., “Task allocation for wide area search munitions with variable path length,” in *American Control Conference, 2003. Proceedings of the 2003*, vol. 4, pp. 3472–3477, IEEE, 2003.
- [106] SHAIKH, M. and CAINES, P., “On trajectory optimization for hybrid systems: Theory and algorithms for fixed schedules,” in *IEEE Conference on Decision and Control*, vol. 2, pp. 1997–1998, IEEE; 1998, 2002.
- [107] SHAIKH, M. and CAINES, P., “On the optimal control of hybrid systems: Optimization of trajectories, switching times, and location schedules,” in *Proceedings of the 6th international conference on Hybrid systems: computation and control*, pp. 466–481, Springer-Verlag, 2003.

- [108] SHAIKH, M. and CAINES, P., “On the hybrid optimal control problem: Theory and algorithms,” *Automatic Control, IEEE Transactions on*, vol. 52, no. 9, pp. 1587–1603, 2007.
- [109] SHUCKER, B., MURPHEY, T., and BENNETT, J., “A method of cooperative control using occasional non-local interactions,” in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1324–1329, IEEE, 2006.
- [110] SMITH, B., HOWARD, A., MCNEW, J., WANG, J., and EGERSTEDT, M., “Multi-robot deployment and coordination with Embedded Graph Grammars,” *Autonomous Robots*, vol. 26, no. 1, pp. 79–98, 2009.
- [111] SMITH, B., WANG, J., EGERSTEDT, M., and HOWARD, A., “Automatic formation deployment of decentralized heterogeneous multi-robot networks with limited sensing capabilities,” in *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*, pp. 730–735, IEEE, 2009.
- [112] SOUTHWOOD, R. and HENDERSON, P., *Ecological methods*. Wiley-Blackwell, 2000.
- [113] STEINBERG, M., “Intelligent autonomy for unmanned naval systems,” in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 6230, p. 34, 2006.
- [114] STONEKING, C., DiBONA, P., and HUGHES, A., “Multi-uav collaborative sensor management for uav team survivability,” tech. rep., Lockheed Martin Advanced Technology Laboratories, 2006.
- [115] TANNER, H., “On the controllability of nearest neighbor interconnections,” in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 3, pp. 2467–2472, IEEE, 2004.
- [116] TANNER, H., JADBABAIE, A., and PAPPAS, G., “Flocking in fixed and switching networks,” *Automatic Control, IEEE Transactions on*, vol. 52, no. 5, pp. 863–868, 2007.
- [117] TANNER, H., PAPPAS, G., and KUMAR, V., “Leader-to-formation stability,” *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 3, pp. 443–455, 2004.
- [118] TOMLIN, C., PAPPAS, G., and SASTRY, S., “Noncooperative conflict resolution [air traffic management],” in *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, vol. 2, pp. 1816–1821, IEEE, 1997.
- [119] TWU, P., CHIPALKATTY, R., DE LA CROIX, J., RAMACHANDRAN, T., SHIVLEY, J., EGERSTEDT, M., RAHMANI, A., and YOUNG, R., “A hardware

- testbed for multi-UAV collaborative ground convoy protection in dynamic environments,” in *Modeling and Simulation Technologies (MST) Conference, 2011 AIAA*, pp. AIAA–2011–6665, AIAA, 2011.
- [120] TWU, P., CHIPALKATTY, R., RAHMANI, A., EGERSTEDT, M., and YOUNG, R., “Air traffic maximization for the terminal phase of flight under FAA’s NextGen framework,” in *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th*, pp. 2–C, IEEE, 2010.
- [121] TWU, P. and EGERSTEDT, M., “Controllability of homogeneous single-leader networks,” *Asian Journal of Control, Submitted for Review in June 2011*.
- [122] TWU, P. and EGERSTEDT, M., “Multi-robot search and rescue: an open-ended educational bridge between theory and practice,” *IEEE Robotics and Automation Magazine, Submitted for Review in Feb. 2011*.
- [123] TWU, P. and EGERSTEDT, M., “Optimal decentralization of multi-agent motions,” in *American Control Conference (ACC), 2010*, pp. 2326–2331, IEEE, 2010.
- [124] TWU, P., EGERSTEDT, M., and MARTINI, S., “Controllability of homogeneous single-leader networks,” in *Proceedings of the 49th IEEE Conference on Decision and Control 2010*, vol. 2, pp. 1538–1543, 2010.
- [125] TWU, P., MARTIN, P., and EGERSTEDT, M., “Graph process specifications for hybrid networked systems,” *Discrete Event Dynamic Systems, Accepted on Feb. 2012*.
- [126] TWU, P., MARTIN, P., and EGERSTEDT, M., “Graph process specifications for hybrid networked systems,” in *Discrete Event Systems, 2010. WODES 2010. 10th International Workshop on*.
- [127] VAUGHAN, R., “Massively multi-robot simulation in stage,” *Swarm Intelligence*, vol. 2, no. 2, pp. 189–208, 2008.
- [128] VERMA, S., LOZITO, S., and TROT, G., “Preliminary guidelines on flight deck procedures for very closely spaced parallel approaches,” *International Council for Aeronautics (ICAS) Anchorage, AK*, 2008.
- [129] VICSEK, T., CZIRÓK, A., BEN-JACOB, E., COHEN, I., and SHOCHET, O., “Novel type of phase transition in a system of self-driven particles,” *Physical Review Letters*, vol. 75, no. 6, pp. 1226–1229, 1995.
- [130] WILDE, J., DIBIASO, D., and NERVEGNA, M., “Team planning for unmanned vehicles in the risk-aware mixed-initiative dynamic replanning system,” in *OCEANS 2007*, pp. 1–8, IEEE, 2007.

- [131] WOLLKIND, S., VALASEK, J., and IOERGER, T., “Automated conflict resolution for air traffic management using cooperative multiagent negotiation,” in *AIAA guidance, navigation, and control conference*, pp. 2004–4992, 2004.
- [132] XIAO, L., BOYD, S., and LALL, S., “A space-time diffusion scheme for peer-to-peer least-squares estimation,” in *Proceedings of the 5th international conference on Information processing in sensor networks*, pp. 168–176, ACM, 2006.
- [133] XU, X. and ANTSAKLIS, P., “Optimal control of switched autonomous systems,” in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 4, pp. 4401–4406, IEEE, 2002.
- [134] XU, X. and ANTSAKLIS, P., “Optimal control of switched systems via nonlinear optimization based on direct differentiations of value functions,” *International Journal of Control*, 75, vol. 16, no. 17, pp. 1406–1426, 2002.

VITA

Philip Y Twu was born on June 9, 1986, in Rockville, MD. He attended the University of Maryland, College Park, in Fall 2004 and graduated in Spring 2008 with a B.S. in Electrical Engineering (with honors) and a B.S. in Computer Science. He received a M.S. in Electrical and Computer Engineering from the Georgia Institute of Technology in Fall 2009, and earned his Ph.D. in Electrical and Computer Engineering under the supervision of Dr. Magnus Egerstedt in the Georgia Robotics and Intelligent Systems (GRITS) Lab. His graduate education consisted of a primary focus in systems and control, with a secondary focus in digital signal processing, and a minor in mathematics.

While at Georgia Tech, Philip was the graduate teaching assistant for two undergraduate courses and also co-taught a graduate-level course on networked control systems. For his efforts, he was awarded the 2009 Outstanding Graduate Assistant Award by the Department of Electrical and Computer Engineering. He also collaborated closely with Rockwell Collins, Inc. on a number of research projects involving air traffic merging and spacing and multi-UAV convoy protection. The resulting paper [120] was awarded the Best Graduate Student Paper Award at the 29th Digital Avionics Systems Conference (DASC).

When not doing research, Philip enjoys working out at the campus recreation center and training in martial arts. Before arriving at Georgia Tech, Philip had prior experience in Wushu, Tae Kwon Do, and Yang Style Tai Chi. Since moving to Atlanta, he has been actively training in Jeet Kune Do, Filipino Kali, and Keysi Fighting Method (KFM). Recently, he has also started taking classes in Gracie Jiu-Jitsu and Judo.