Algorithms for Timing and Sequencing Behaviors in Robotic Swarms

Sasanka Nagavalli

CMU-RI-TR-18-19

May 2018

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Katia Sycara, Chair, *Carnegie Mellon University*
Howie Choset, *Carnegie Mellon University*
Maxim Likhachev, *Carnegie Mellon University*
Nilanjan Chakraborty, *Stony Brook University*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Keywords: Robotic Swarms, Behavior Composition, Human-Swarm Interaction

For my parents and my sister

Abstract

Robotic swarms are multi-robot systems whose global behavior emerges from local interactions between individual robots and spatially proximal neighboring robots. Each robot can be programmed with several local control laws that can be activated depending on an operator's choice of global swarm behavior (e.g. flocking, aggregation, formation control, area coverage). In contrast to other multi-robot systems, robotic swarms are inherently scalable since they are robust to addition and removal of members with minimal system reconfiguration. This makes them ideal for applications such as search and rescue, environmental exploration and surveillance.

Practical missions often require a combination of swarm behaviors and may have dynamically changing mission goals. However, a robotic swarm is a complex distributed dynamical system, so its state evolution depends on the timing as well as sequence of the supervisory inputs. Thus, it is difficult to predict the effects of an input on the state evolution of the swarm. More specifically, after becoming aware of a change in mission goals, it is unclear at what time a supervisory operator must convey this information to the swarm or which combination of behaviors to use to accomplish the new goals.

The main challenges we address in this thesis are characterizing the effects of input timing on swarm performance and using this theory to inform automated composition of swarm behaviors to accomplish updated mission goals.

We begin by formalizing the notion of Neglect Benevolence — the idea that delaying the application of an input can sometimes be beneficial to overall swarm performance — and using the developed theory to demonstrate experimentally that humans can learn to approximate optimal input timing. In an adversarial setting, we also demonstrate that by altering only the timing of consensus updates for a subset of the swarm, we can influence the agreement point of the entire swarm.

Given a library of swarm behaviors, automated behavior composition consists of identifying a behavior schedule that must specify (1) the appropriate sequence of behaviors and (2) the corresponding duration of execution for each behavior. Applying our notion of Neglect Benevolence, it is clear these two parts are intricately interdependent. By first assuming the durations are known, we present an algorithm to identify the optimal behavior sequence to achieve a desired swarm mission goal when our library contains general swarm behaviors. By restricting our library to consensus-based swarm behaviors, we then relax the assumption on known durations and present an algorithm to simultaneously find the sequence and durations of swarm behaviors to time-optimally accomplish multiple unordered goals.

Acknowledgments

First and foremost, I would like to thank my advisor, Professor Katia Sycara. I am very grateful to have had her as my guide on this journey, during which I had many opportunities to learn and grow. Our discussions have been invaluable in shaping my research and without her advice and support, this thesis would not have been possible.

I would like to thank my thesis committee members. I appreciate Professor Howie Choset's expertise, enthusiasm and words of encouragement, which inspired me to pursue this endeavor. I appreciate Professor Maxim Likhachev's experience, insight and helpful comments. Throughout my graduate research, Professor Nilanjan Chakraborty has always been a mentor to me and our discussions have always steered my research in fruitful new directions.

I am very thankful to Professor Michael Lewis for discussions and advice during our lab meetings. The experiments involving human participants were possible as a result of his guidance. Thank you also to Dr. Changjoo Nam for comments during lab meetings. Thank you to Tony Dear for very insightful discussions early in my research.

I have been fortunate to have had many great labmates over the years, who I thank in the order we met, including Lingzhi Luo, Phillip Walker, Shih-Yi Chien, Robert Thome, Wenhao Luo, Shehzaman Khatib, Anqi Li, Navyata Sanghvi, Meghan Chandarana, Ramitha Sundar, Jaeho Bang, Kyle Morris, Gabriel Arpino, Huao Li, Akshat Agarwal, Sha Yi, Sumit Kumar, Vigneshram Krishnamoorthy, Yifan Ding, Xinzhi Wang and Fan Jia. Special thanks to Wenhao Luo for many interesting discussions over delicious food.

I am thankful to Suzanne Muth for graduate program related support during my time in the Robotics Institute.

Finally, I am deeply grateful to my parents and my sister for their boundless support and encouragement, which enabled me to complete this endeavor.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Robotics is a vast field that was traditionally characterized by the use of a single complex robot applying perception, cognition and action to complete a mission. Growing capabilities of individual robots, a better understanding of individual limitations and a healthy surge in envisioned applications requiring multiple co-operating robots led to interest in multi-robot systems. Multi-robot teams can complete complex missions that would not be possible with an individual robot (e.g. patrolling, environmental monitoring) and can complete other missions more effectively and efficiently than individual robots operating independently (e.g. search and rescue). More recently, the need for scalability and robustness to individual failure has led to interest in a particular class of multi-robot system known as a *robotic swarm*. Robotic swarms are distinguished from other multi-robot systems by the fact that their global behavior emerges from local interactions between robots and objects within their spatial neighborhood (e.g. nearby robots, environmental factors, terrain, obstacles), communication neighborhood (i.e. robots with which they can communicate) or sensing neighborhood (e.g. robots, objects or events they can perceive). These emergent global behaviors may be biologically inspired (e.g. flocking, aggregation) or engineered (e.g. formation generation and maintenance). Ideally, individual robots may be added or removed from the swarm with minimal reconfiguration, which indicates that a robotic swarm is a multi-robot system with the potential to be scalable and robust to individual failure.

In many realistic scenarios, mission goals change intermittently or arise dynamically, necessitating supervisory interaction with the robotic swarm. Supervisory interaction can take many forms including both discrete commands (e.g. selecting a rendezvous point, selecting a leader, selecting a collective behavior) or continuous signals (e.g. applying a time-varying virtual force)

or some combination of discrete and continuous signals (e.g. selecting a leader and driving it to a goal location while having the other swarm members follow via a flocking behavior). Sometimes it is useful to think of a continuous signal applied over a known duration as a discrete command moving the swarm from a known initial state at the beginning of the duration to a final state at the end of the duration. In this way, the final state is a function of both the discrete command and the duration of application. This final approach is the one taken in this thesis. Our discrete commands are swarm behaviors selected by a supervisory operator and the operator may switch transiently between behaviors after applying them for a particular duration.

In this thesis, we study swarm behavior composition in interaction between a supervisory operator and a robotic swarm to improve the overall performance of the system. Behavior composition consists of (a) identifying the appropriate sequence of swarm behaviors and (b) the times at which to switch between behaviors in the sequence. It is then natural to ask *which swarm behaviors should the operator select and for what durations should they be applied to optimize a performance criterion*? We refer to this as the swarm behavior composition problem. The two parts of this problem are intricately interdependent. We begin by identifying approaches to solve each part while holding the other fixed (i.e. identify switch times given sequence or identify sequence given switch times) and then follow with an approach to solve both problems simultaneously, a process we refer to as behavior scheduling.

1.2 Thesis Statement

Swarm behavior composition, consisting of sequencing and switching transiently between swarm behaviors at the appropriate times is a novel approach enabling effective supervisory interaction with robotic swarms for improved joint performance.

1.3 Contributions

This dissertation makes the following contributions.

1. Formally model the swarm behavior composition problem.
2. Characterize the impact of input timing on swarm performance.
3. Given a library of swarm behaviors, develop automated algorithms to optimally find the sequence of swarm behaviors and the durations for which they should be applied.

1.4 Outline of Thesis

This thesis approaches the problem of swarm behavior composition in two parts. We first begin with a thorough investigation of the impact of timing on swarm performance, which we follow with methods for optimal sequencing and scheduling of swarm behaviors.

In **Chapter 2**, we introduce the Neglect Benevolence phenomenon, which is the notion that delaying a supervisory input to the swarm can sometimes improve swarm performance. We formalize this notion in a control theoretic framework and present a proof that all linear time-invariant systems exhibit Neglect Benevolence. We then develop a simple algorithm to identify optimal input times and apply it to find the optimal input time to inform the swarm to switch between two configurations. The work in this chapter has been published in [57].

In **Chapter 3**, we use the formal notion of Neglect Benevolence introduced in the previous chapter to develop a base human-swarm interaction task in which we can empirically study human ability to approximate optimal input times. Characterizing timing of human inputs as redirecting swarm trajectories through state space, we measure human ability to find the correct input time to switch between swarm configurations and compare it to the optimal time identified by our algorithm. We develop a simple visual aid to augment the human interface and measure the human performance with and without the aid. We conclude that human performance is affected by multiple factors, but tends to improve with experience and their approximation of optimal input timing can be significantly improved by the use of a visual aid. The work in this chapter has been published in [58].

In **Chapter 4**, we show that an adversary can apply timing independently of any particular input to influence a swarm performing consensus. By considering a slightly different model of a swarm, where individual agents periodically apply the consensus update rule with different frequencies and delays, we show that by changing only the periods and delays of a subset of swarm members, we can influence the final agreement point of the entire swarm. Some of the work in this chapter has been published in [61].

In **Chapter 5**, given a library of swarm behaviors, we develop a method to find the appropriate sequence of swarm behaviors for a particular task for which none of the behaviors in the library may have been individually designed. Given a library of concrete swarm behaviors, a set of desired goal states defining the task objective, a performance criterion to evaluate solution quality and known switch times, we develop an informed search algorithm that finds the appropriate behavior sequence to complete the task with minimum cost. The algorithm is proven to be optimal and complete. We demonstrate the utility of the algorithm in two different applications: swarm navigation and swarm dynamic area coverage. The work in this chapter has been published

in [59].

In **Chapter 6**, we relax the assumption that switch times must be known and develop a method to schedule (i.e. simultaneously find the sequence and duration of application) swarm behaviors to achieve an unordered set of intermediate goals in minimum time. Our algorithm is proven to be locally optimal in the durations and the sequence in which the goals are achieved has bounded suboptimality. We revisit the problem of swarm configuration control and apply our algorithm to find the optimal schedule. The work in this chapter has been published in [60].

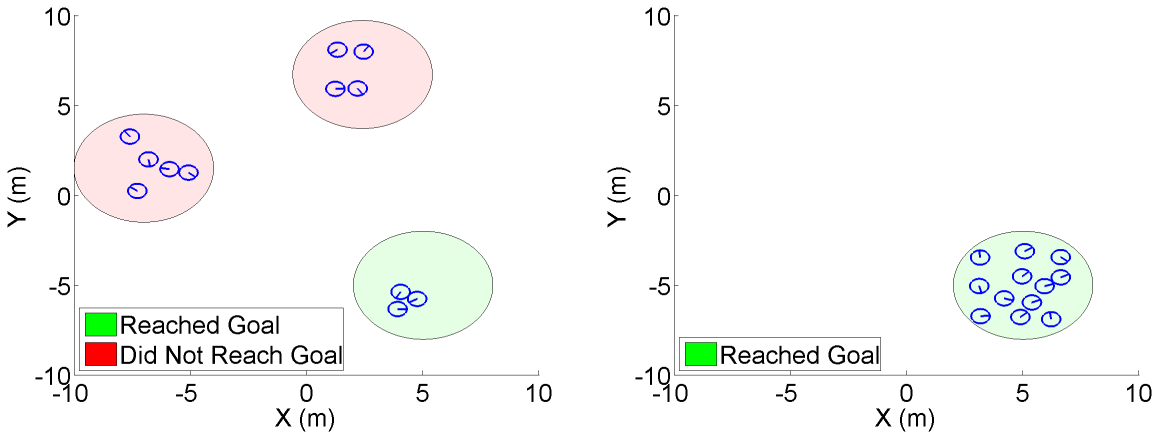
Finally, in **Chapter 7**, we present a summary of our contributions and present some directions for future work.

Chapter 2

Neglect Benevolence in Human Control of Robotic Swarms

Robotic swarms use control laws based on local information about the environment and/or other members of the swarm within their spatial neighborhood to achieve various collective behaviors. The key advantage of swarms is that the behaviors generated are robust to individual robot failures. Practical missions or tasks for robotic systems may require a combination of different swarm behaviors in dynamic environments or situations. However, swarms are not yet capable of performing such complex dynamic missions autonomously. Human experiments with a simulated swarm system performing a variety of tasks including environment exploration [45] and radiation source detection [7] have been performed. In a previous experimental study in a foraging scenario [79], we found that the performance of the system was affected by the time between two commands that the human was applying to the robots. In particular, we found that one group of subjects who performed well waited for some time after they issued a command before issuing another corrective command (when they wanted to change the direction in which the swarm was heading). We termed the phenomenon *Neglect Benevolence*, since neglecting the swarm (or not "correcting" the swarm) for some amount of time leads to better performance of the system. The goal of this chapter is to formalize the notion of Neglect Benevolence.

Our human experiment findings indicated that when the swarm was in a transient state (i.e. moving towards one goal), application of another input (that changes the goal) could have different effects depending on the timing of the inputs. To verify this in a more controlled setting we set up a simulation of a small swarm system (12 robots) performing rendezvous. The robots move under simple control laws given in [36] with a repulsive potential to avoid inter-robot collision and an attractive potential field to maintain swarm cohesion. The operator inputs change the



(a) When the human input is applied at $t = 0$ the swarm splits and only 3 robots reach the goal

(b) When the human input is delayed all the robots reached the goal

Figure 2.1: Effect of Different Human Control Input Times on the Final State of the Swarm

point at which the agents rendezvous. Figure 2.1 shows an example of the difference in the end result when the input is applied at two different times. In Figure 2.1a when the input is applied immediately after the operator wants to change the goal point of the rendezvous, the robot swarm splits and some of the robots, shown in the figure via the pink circles, wander off never reaching the operator goal, and only a small number (3) of robots, shown in the green circle, reaches the goal. However, when there is a delay in applying the input, shown in Figure 2.1b, the swarm stays as a single entity and converges around the operator goal. This example further reinforces the experimental findings that for a human to influence the operations of a robotic swarm the timing of the human input can greatly affect the performance of the robotic system. We performed additional simulations with large numbers of robots under different control laws and initial configurations and found similar results in that the timing of the operator input affected the system performance.

Since the behavior of a swarm system is emergent via the robots' autonomous interactions, it is difficult for humans to have an intuitive understanding of the evolution of the swarm states. Consequently, it is difficult for humans to have a reasonable guess of the appropriate delay after which to apply the input. Therefore, formal characterization of the notion of Neglect Benevolence and methods to calculate the best time the operator must give an input are needed. These results are of practical significance since they can enable the design of computational aids for the human to ensure that the human input is given to the swarm at the appropriate time.

Note that for the results shown in Figure 2.1, the underlying dynamics of the swarm system is nonlinear and the results show that the inappropriate timing of the human input can make the

system unstable. However, the system does not need to be nonlinear for Neglect Benevolence to be present. Even if (a) the system dynamics is linear and (b) the swarm is always guaranteed to be stable, which is the case with well-designed linear systems, Neglect Benevolence can still exist (as we will show later in the chapter).

Different performance measures for the swarm system can be used for defining Neglect Benevolence. Examples include the time taken by the swarm to reach the human goal and the energy used by the swarm to reach the goal. We will study Neglect Benevolence for a particular class of linear systems, namely linear systems with *normal* dynamics matrix, where the performance measure is defined as the time taken by the swarm to reach the human specified goal. The dynamics matrix of a swarm that performs consensus-based behaviors falls within this class. Consensus-based behaviors include rendezvous and configuration control. The human input can control the inter-robot spacing and thereby control the final shape into which the robots converge. We will illustrate our concepts with this example (see Section 2.4). We make the following contributions in this chapter: (a) formally define the new notion of Neglect Benevolence, (b) prove the existence of Neglect Benevolence for a set of linear dynamical systems, (c) provide an analytic characterization and an algorithm for calculating the optimal input time, (d) apply the analysis to human control of swarm configuration to illustrate the approach.

This chapter is organized as follows. In Section 2.1, we present a discussion of the related work. In Section 2.2, we formalize the notion of Neglect Benevolence and prove the existence of Neglect Benevolence for a class of linear systems. In Section 2.3, we present a procedure to compute the delay time efficiently for linear systems with a normal dynamics matrix. In Section 2.4, we show an application of the analysis for configuration control of swarms. Finally, in Section 2.5, we present our simulation results.

2.1 Related Work

Robotic swarms, where local interactions between robots produce a variety of emergent collective behaviors have been extensively studied. Swarms can achieve behaviors such as flocking [14, 24, 69, 74], rendezvous [16], deployment [20], and foraging [7, 30]. Formation control of robotic swarms has been proposed to adjust the spatial configuration of multi-robotic systems so that they satisfy the requirements of deployment tasks or they can adapt to different environments while performing tasks [5, 8, 31, 77].

However, schemes that include human operator in the loop together with robotic swarms are required for some complicated tasks (e.g. complex surveillance and reconnaissance). For using

swarm robotic systems in human-supervised missions, human swarm interaction (HSI) has been studied in the extant literature [22, 27, 34, 37, 42, 45]. The concept of Neglect Tolerance has been introduced to capture the idea that a human operator can neglect robots that work independently of one another in a multi-robot system for a certain time before system performance degrades [25, 65, 66, 75]. One key issue in neglect tolerance is how to schedule the attention of the human operator among the multiple and independent robots so that the neglect time between servicing robots is minimized [19, 52, 54, 85]. In Neglect Tolerance, it is assumed that the robot system's performance will degrade over time and the system will need human interaction to restore performance to a desirable level. In contrast, the concept of Neglect Benevolence is proposed in human swarm interaction to capture the idea that it may be beneficial for the human operators to wait for a certain length of time before applying input to the system while the swarm evolves to a stable state [79]. The concept of Neglect Benevolence has not been quantitatively characterized to date. Based on the observation of Neglect Benevolence in human control of robotic swarms, we propose a formal way to characterize Neglect Benevolence under different performance criteria by considering the swarm system dynamics.

2.2 Formalizing the Notion of Neglect Benevolence

Consider a system that consists of a swarm of robots guided by a human operator. The robot swarm can be modeled as an autonomous distributed dynamical system that is capable of incorporating an input from an intelligent external controller (i.e. human operator). Each individual component of the distributed dynamical system has a state $\mathbf{x}^{(i)}$ that evolves as a function of time t. For a system with m components, the joint state $\mathbf{x}$ of the entire dynamical system (i.e. swarm) is given by concatenating the states of the individual components (i.e. robots). The set of all states of the system is known as the state space $\mathcal{X}$. The state $\mathbf{x} \in \mathcal{X}$ of the entire system evolves over time according to the system dynamics f, which relate the change in the state over time $\dot{\mathbf{x}}$ to the current state $\mathbf{x}$ and autonomous input $\mathbf{u}_a \in \mathcal{U}_a$, where $\mathcal{U}_a$ is a bounded set. Each robot in the system contains an automatic controller $g^{(i)}$ that computes the autonomous input $\mathbf{u}_a^{(i)}$ to the robot's actuators based on an estimate $\hat{\mathbf{x}}$ of the (partial) state of the system and the human input $\mathbf{u}_h$. The human has a desired — possibly time-varying — set of goals that he/she would like to achieve with the robot swarm and a fixed set of inputs $\mathcal{U}_h$ that they can apply to influence the swarm. The goals may arise dynamically due to unexpected events in the environment or mission changes and are not known in advance. The goals can be expressed as a set of goal states $\mathcal{X}_g \subset \mathcal{X}$. The human combines this knowledge of desired goal states $\mathcal{X}_g$ with their possibly noisy, incorrect or incomplete knowledge $\tilde{\mathbf{x}}$ of the swarm's current state via a process h to

generate an input $\mathbf{u}_h \in \mathcal{U}_h$ to the robot swarm. The relationships between all of these variables are summarized in the equations below.

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{x}^{(1)}(t)^T & \mathbf{x}^{(2)}(t)^T & \cdots & \mathbf{x}^{(m)}(t)^T \end{bmatrix}^T \tag{2.1}$$

$$\mathbf{u}_a(t) = \begin{bmatrix} \mathbf{u}_a^{(1)}(t)^T & \mathbf{u}_a^{(2)}(t)^T & \cdots & \mathbf{u}_a^{(m)}(t)^T \end{bmatrix}^T \tag{2.2}$$

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}_a(t)) \tag{2.3}$$

$$\mathbf{u}_a^{(i)}(t) = g^{(i)}(\hat{\mathbf{x}}(t), \mathbf{u}_h(t)) \tag{2.4}$$

$$\mathbf{u}_h(t) = h(\tilde{\mathbf{x}}(t), \mathcal{X}_g) \tag{2.5}$$

For our purposes, we assume that the human can come up with the input $\mathbf{u}_h$ without modeling the function h. Furthermore, we will assume that each robot in the swarm has perfect access to its own and its neighbors' state (i.e. $\hat{\mathbf{x}} = \mathbf{x}$). The set of stable equilibrium points $\mathcal{X}_n \subset \mathcal{X}$ of the system is called the **natural goal** set. When there is no human influence (i.e. $\mathbf{u}_h = \mathbf{0}$), any control input $\mathbf{u}_n \in \mathcal{U}_a$ that drives the system to the natural goal set is called a **natural input**.

A path through the state space is a sequence of states. The path taken by a system from any initial state $\mathbf{x}_s$ towards a natural goal, when $\mathbf{u}_h = \mathbf{0}$, is called the **natural path**.

As the system proceeds towards its natural goal $\mathbf{x}_n$, the human may wish to apply an input $\mathbf{u}_h$ to influence the swarm towards the desired set of goal states $\mathcal{X}_g$. The input may be instantaneous or persistent, but for illustration, assume that there is only one opportunity to apply or activate it. Given a performance criterion that the human wishes to maximize, if applying the input at some time $T > 0$ results in better performance than applying the input at any time $t < T$, then all states of the system during time $t < T$ are said to be **Neglect Benevolent**. Given a system, a performance criterion, an input $\mathbf{u}_h \in \mathcal{U}_h$ and a set of desired goal states $\mathcal{X}_g$, there may exist a set of Neglect Benevolent states $\mathcal{X}_{nb} \subset \mathcal{X}$ where delaying the application of the input will result in improved system performance. Specifically, when the system begins in state $\mathbf{x}_{nb} \in \mathcal{X}_{nb}$ and follows a natural path through the state space, there is some future state at which applying the input results in better performance than applying the input at state $\mathbf{x}_{nb}$. Systems where $\mathcal{X}_{nb} \neq \varnothing$ are said to exhibit the property of **Neglect Benevolence**. The performance criterion to be used is task-specific and depends on the mission goals. Examples of performance criteria include the time taken to reach the human goal and the total energy required to reach the human goal. For the rest of the chapter, we will use the time required by the system to reach the human goal as the performance criterion.

2.3 Analysis of Neglect Benevolence in Linear Time Invariant Systems

In this section, we quantitatively analyze the concept of Neglect Benevolence in human control of robot swarm systems. In Section 2.4, we use the analysis to determine Neglect Benevolence and best time to apply human input for swarm configuration control. We focus on linear time invariant (LTI) robot swarm systems with a normal dynamics matrix. In Section 2.3.1, we first consider a swarm system without external human input, and numerically solve the system convergence time based on the eigenvalue decomposition. Then we discuss the situation when the human operator wants to give a new command to the dynamic swarm system, and present an algorithmic way of finding the optimal time to impose the human operator's new input so that the time to reach the new goal is minimized. In Section 2.4, we analyze an application of human control in robot swarm configuration control. We first design a control law for a robot swarm system so that it can form desired spatial configurations. The swarm system under the designed control law has a symmetric dynamics matrix, and we apply the above analysis results to optimize the configuration control of the system.

2.3.1 Neglect Benevolence in LTI Systems with a Normal Dynamics Matrix

Consider a linear time-invariant (LTI) robot swarm system with a normal dynamics matrix and without external human inputs.

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) \tag{2.6}$$

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{2.7}$$

where $\mathbf{A}$ is the dynamics matrix, $\mathbf{x}(t)$ represents the system state vector at time t and $\mathbf{x}_0$ is the initial state. We assume that $\mathbf{A}$ is normal (i.e. $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T$) and all its eigenvalues have negative real parts (so that the system is stable). Below we first numerically solve the system convergence time based on the eigenvalues of $\mathbf{A} + \mathbf{A}^T$.

Solving the dynamic equation, we get

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}_0 \tag{2.8}$$

Since $\mathbf{A}$ is normal, $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T$, so

$$\|\mathbf{x}(t)\|_2^2 = \mathbf{x}_0^T\mathbf{Q}e^{\mathbf{\Lambda}t}\mathbf{Q}^T\mathbf{x}_0 \tag{2.9}$$

where $\mathbf{A}^T + \mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ is the eigenvalue decomposition of the symmetric matrix $\left(\mathbf{A}^T + \mathbf{A}\right)$ and $\mathbf{\Lambda}$ is a diagonal matrix with elements λ_i, where $\lambda_i \in \mathbb{R}$ are the eigenvalues of $\left(\mathbf{A}^T + \mathbf{A}\right)$. Define $\mathbf{y} = \mathbf{Q}^T\mathbf{x}_0$, we have

$$\|\mathbf{x}(t)\|_2^2 = \mathbf{y}^T e^{\mathbf{\Lambda} t}\mathbf{y} = \sum y_i^2 e^{\lambda_i t} \tag{2.10}$$

From the above, the 2-norm of the state evolution function is a sum of decaying exponentials with positive coefficients. We can find an upper bound on the convergence time t_{natural} for the system without external input using the maximum eigenvalues. Define $\alpha = \max_i \lambda_i$ and ε as the tolerance for convergence.

$$\|\mathbf{x}(t_{\text{natural}})\|_2^2 = \sum y_i^2 e^{\lambda_i t_{\text{natural}}} \leq \mathbf{y}^T\mathbf{y} e^{\alpha t_{\text{natural}}} < \varepsilon^2 \tag{2.11}$$

$$t_{\text{natural}} \leq \frac{1}{\alpha}\ln\frac{\varepsilon^2}{\mathbf{y}^T\mathbf{y}} = \frac{1}{\alpha}\ln\frac{\varepsilon^2}{\mathbf{x}_0^T\mathbf{x}_0} \tag{2.12}$$

Define $f(t) = \|e^{\mathbf{A}t}\mathbf{x}_0\|_2^2$ and $F(t) = f(t) - \varepsilon$. Since $f(t)$ is monotonically decreasing and converges to 0, $F(t)$ only has one root, which represents the convergence time of the system. A numerical solution for the convergence time t_{natural} can be found using the following iterative Newton's Method where $F'(t) = \sum_{i=1}^{n} y_i^2 \lambda_i e^{\lambda_i t}$.

$$t_{k+1} = t_k - \frac{F(t)}{F'(t)} \tag{2.13}$$

Now consider the situation when the human operator wants to give a new input to the dynamic swarm system.

$$\dot{\mathbf{x}}(t) = \mathbf{A}\left(\mathbf{x}(t) - K\theta\left(t - t_{\text{input}}\right)\right) \tag{2.14}$$

where K represents the human input, so that the system now converges to state K rather than the origin. Here $\theta(t)$ is the Heaviside step function such that $\theta(t) = 0$ for $t < 0$ and $\theta(t) = 1$ for $t \geq 0$. t_{input} is the time when the new input K is imposed on the system. Solving Equation (2.14), we have

$$\mathbf{x}(t) = e^{\mathbf{A}t}\left(\mathbf{x}_0 - e^{-\mathbf{A}t_{\text{input}}}K\theta\left(t - t_{\text{input}}\right)\right) + K\theta\left(t - t_{\text{input}}\right) \tag{2.15}$$

For a stable system, $e^{\mathbf{A}t}$ is a decaying matrix exponential, so the system will converge to the input K at some time t_{goal} after t_{input}.

$$\mathbf{x}(t_{\text{goal}}) = e^{\mathbf{A}t_{\text{goal}}}\left(\mathbf{x}_0 - e^{-\mathbf{A}t_{\text{input}}}K\right) + K \tag{2.16}$$

Define $\mathbf{x}_d = \mathbf{x}_0 - e^{-\mathbf{A}t_{\text{input}}}K$,

$$\mathbf{x}\left(t_{\text{goal}}\right) = e^{\mathbf{A}t_{\text{goal}}}\mathbf{x}_d + K \tag{2.17}$$

$$\left\|e^{\mathbf{A}t_{\text{goal}}}\mathbf{x}_d\right\|_2 < \varepsilon \tag{2.18}$$

So we can solve t_{goal} in the same way as we solve for t_{natural}. The only difference is that the initial condition changes from $\mathbf{x}_0$ to $\mathbf{x}_d$.

The convergence time t_{goal} depends on t_{input} (implicitly included in the expression of $\mathbf{x}_d$). We can use the following naive algorithm to sample all possible $t_{\text{input}} \leq t_{\text{natural}}$ and find the best input time t^*_{input} so that t_{goal} is minimized:

1. Solve the dynamics equations (i.e. find $\mathbf{x}\left(t\right)$).
2. Find the convergence time to the natural goal without external input.

$$t_{\text{natural}} = \min\left\{t_f \mid \forall t \geqslant t_f : \left\|e^{\mathbf{A}t}\mathbf{x}_0\right\|_2 < \varepsilon\right\}$$

3. For each possible input time $t_{\text{input}} \leqslant t_{\text{natural}}$, compute the convergence time to the new input goal.

$$t_{\text{goal}} = \min\left\{t_f \mid \forall t \geq t_f : \|e^{\mathbf{A}t}(\mathbf{x}_0 - e^{-\mathbf{A}t_{\text{input}}}K)\|_2 < \varepsilon\right\}$$

4. The input time, t^*_{input}, that results in the minimum convergence time to the new input goal, t_{goal}, is the optimal input time.

Whenever $t^*_{\text{input}} > 0$, it means that the new input should be delayed for t^*_{input} before being imposed on the system, so that the convergence performance of the system is optimized. Generally speaking, whether $t^*_{\text{input}} = 0$ or $t^*_{\text{input}} > 0$ would depend on the dynamics matrix $\mathbf{A}$, the natural goal, the newly input goal K, as well as the initial state $\mathbf{x}_0$. However, for any dynamics matrix $\mathbf{A}$ (normal and exponentially stable), we can construct a condition (including the new goal K, initial state $\mathbf{x}_0$), such that under such a condition, $\mathbf{x}_0$ is always Neglect Benevolent, and thus $t^*_{\text{input}} > 0$ (See Theorem 1). The above analysis provides a quantitative way to characterize Neglect Benevolence.

Theorem 1. *Consider an exponentially stable linear system where the human input is incorporated using Equation 2.14.*

$$\dot{\mathbf{x}}\left(t\right) = \mathbf{A}\left(\mathbf{x}\left(t\right) - K\theta\left(t - t_{input}\right)\right)$$

In such a system, for any initial state $\mathbf{x}_0$, if the state K lies along the natural path from $\mathbf{x}_0$ to the origin, then the state $\mathbf{x}_0$ must be Neglect Benevolent.

Proof. Let the system take time t_1 to go from initial state $\mathbf{x}_0$ to the state K along the natural path under the dynamics $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t)$. Without loss of generality, we can assume that there is a state $\mathbf{x}_1$ along the natural path such that the time taken to goal from state K to state $\mathbf{x}_1$ is t_1. Define $\mathbf{x}_2 = K - \mathbf{x}_1$. Suppose that it takes time t_2 from $\mathbf{x}_0$ to K under the new input dynamics $\dot{\mathbf{x}}(t) = \mathbf{A}(\mathbf{x}(t) - K)$.

$$e^{\mathbf{A}t_1}\mathbf{x}_0 = K \tag{2.19}$$

$$\|e^{\mathbf{A}t_2}(\mathbf{x}_0 - K)\|_2 < \varepsilon \tag{2.20}$$

Substituting Equation 2.19 to Equation 2.20, we have

$$\|e^{\mathbf{A}(t_2-t_1)}\left(I - e^{\mathbf{A}t_1}\right)K\|_2 < \varepsilon \tag{2.21}$$

$$\|e^{\mathbf{A}(t_2-t_1)}\mathbf{x}_2\|_2 < \varepsilon \tag{2.22}$$

Thus, the time taken to go from state $\mathbf{x}_2$ to the origin along the natural path is $t_2 - t_1$, which must be positive. So $t_2 > t_1$, which means that it takes longer when $t_{\text{input}} = 0$ than $t_{\text{input}} = t_1$ for the system to reach the state K from initial state $\mathbf{x}_0$. So $\mathbf{x}_0$ must be Neglect Benevolent. □

In Theorem 1, we showed that for any input K, we can find at least one initial state $\mathbf{x}_0$ that must be Neglect Benevolent. Conversely, it can be shown via similar arguments that given any initial state $\mathbf{x}_0$, we can find at least one input K under which that state $\mathbf{x}_0$ is Neglect Benevolent. Thus, for any exponentially stable linear system, when the set of human inputs $\mathcal{U}_h$ is non-empty, the set of of Neglect Benevolent states $\mathcal{X}_{nb}$ is non-empty. Therefore, any exponentially stable linear system exhibits Neglect Benevolence.

2.4 Application to Robot Swarm Configuration Control

In this section, we use the above quantitative approach to analyze Neglect Benevolence in human control of robot swarm configurations. We first design a control law for a robot swarm system that enables it to form various spatial configurations. The swarm system under the designed control law has a symmetric (normal) dynamics matrix. Consequently, we can apply the analysis results presented in the previous section to optimize the configuration control of the system.

Consider a swarm of robots where each robot has a bidirectional communication link with some subset of the other robots in the swarm. Assume that the communication topology of robots can be modeled by an undirected connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex is a robot and each edge is a bidirectional communication link. The adjacency matrix for the graph is given by

$\mathbf{A}_{\text{adjacency}} = [a_{ij}]$, where $a_{ii} = 0$ and $a_{ij} \geq 0$ is the weight of the edge between vertex i and vertex j in the graph. If such an edge does not exist in the graph, $a_{ij} = 0$. Note that in an undirected graph, the adjacency matrix is symmetric. The Laplacian matrix is given by $\mathbf{L} = [l_{ij}]$, where $l_{ii} = \sum_j a_{ij}$ and $l_{ij} = -a_{ij}$. Note that when the graph is connected, $\mathbf{L}$ is a singular matrix with one eigenvalue equal to 0 and the vector of all components 1 as the corresponding eigenvector.

$$\mathbf{1}^\top \mathbf{L} = \mathbf{L}\mathbf{1} = \mathbf{0}$$

As presented in [68], the state update equation for the swarm is

$$\dot{\mathbf{x}}(t) = -\mathbf{L}\mathbf{x}(t) \tag{2.23}$$

where the vector $\mathbf{x}(t)$ represents the state of the system at time t, and is a concatenation of all the state values $x_i(t)$ of each robot i in the system. The system would converge to a state $\mathbf{x}_{\text{final}}$, where

$$\mathbf{x}_{\text{final}} = \frac{\sum_i x_i(0)}{n}\mathbf{1} \tag{2.24}$$

Each robot has the same state value, which is the average of all robots' initial states. If the state of each robot is multi-dimensional (e.g. x-positions, y-positions) then Equation 2.23 holds for each dimension. This means that if the state represents the spatial location of robots, then all robots would rendezvous under the state update (or control) law.

Below we design another control law for a connected robot swarm system, so that all robots would rendezvous but in a desired spatial formation around the centroid of their initial location, specified beforehand by a vector parameter K.

Lemma 1. *Under the following control law:*

$$\dot{\mathbf{x}}(t) = -\mathbf{L}(\mathbf{x}(t) - K) \tag{2.25}$$

the robot swarm state would converge to $\mathbf{x}_{final}$, with individual robot's state difference specified by K (i.e. the robot swarm would rendezvous to a desired spatial formation specified by K), around the initial centroid.

Proof. First, we show that the sum of all robots' state values is always a constant.

$$\sum_{i=1}^{n} \dot{x}_i(t) = \mathbf{1}^\top \dot{\mathbf{x}}(t) = -\mathbf{1}^\top \mathbf{L}(\mathbf{x}(t) - K) = 0 \tag{2.26}$$

$$\sum_{i=1}^{n} x_{\text{final},i} = \sum_{i=1}^{n} x_i(0) \tag{2.27}$$

where $x_{\text{final},i}$ represents the final state of robot i.

Next we show that when the state of all robots converges, the difference among their state values is specified by K.

$$\dot{\mathbf{x}}_{\text{final}} = 0 \Rightarrow \mathbf{L}\left(\mathbf{x}_{\text{final}} - K\right) = 0 \tag{2.28}$$

Since we assume that $\mathcal{G}$ is connected, the eigenvector of $\mathbf{L}$ corresponding to eigenvalue 0 is $a\mathbf{1}$, where $a \in \mathbb{R}$.

$$\mathbf{x}_{\text{final}} = K + a\mathbf{1} \tag{2.29}$$

Combining Equation (2.27) and (2.29), we have that

$$x_{final,i} = \frac{\sum_i x_i\left(0\right)}{n} + \left(K_i - \frac{\sum_i K_i}{n}\right) \tag{2.30}$$

So the centroid of the convergence state is the same as the initial centroid, but robots have a spatial formation, specified by K. □

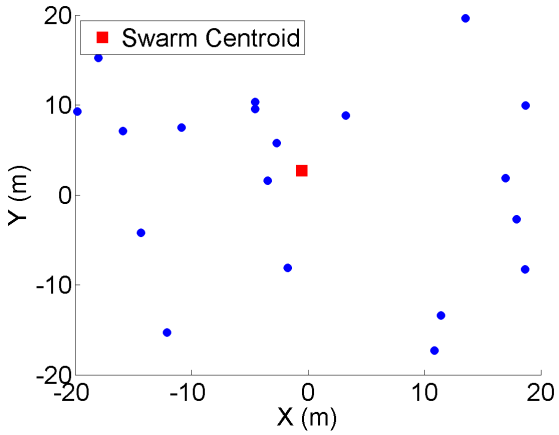
Now if we consider the situation when the human operator intends to give a new input K_2 after the initial K_1, depending on the input time t_{input} of K_2, the control law is

$$\dot{\mathbf{x}}\left(t\right) = -\mathbf{L}\left(\mathbf{x}\left(t\right) - K_1 + \left(K_1 - K_2\right)\theta\left(t - t_{\text{input}}\right)\right) \tag{2.31}$$

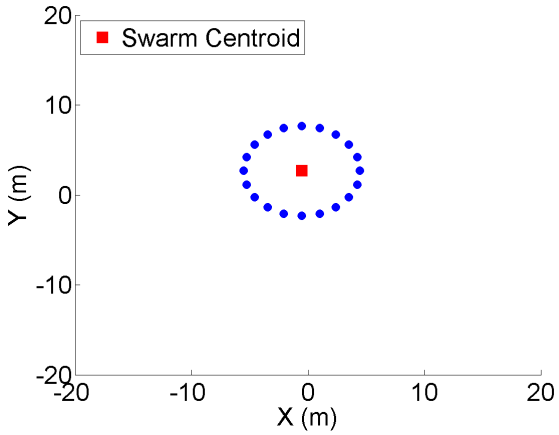
We want to impose the new input K_2 at an optimal time t^*_{input} so that the transient time to the formation specified by K_2 is minimized. Since the dynamics matrix $-\mathbf{L}$ is symmetric, it must be normal. So the system can be viewed as a special case of systems discussed in Section 2.3.1. So we can apply the results in Section 2.3.1 to analyze Neglect Benevolence for robot swarm configuration control.

2.5 Simulation Results for Robot Swarm Configuration Control

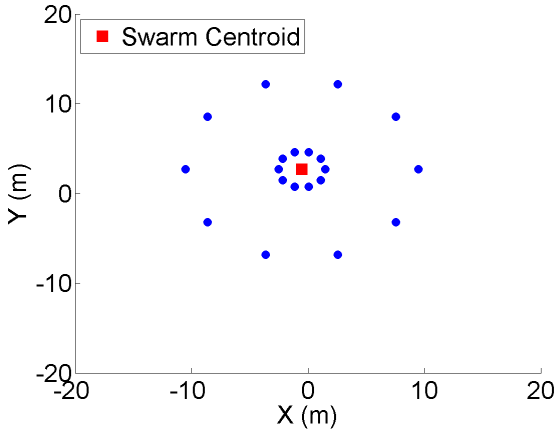
The method of generating various robot configurations presented in the previous section can be used to generate rigid formations for the robots as shown in Figure 2.2. If the state vector $\mathbf{x}$ represents the concatenated x-positions and y-positions of the robots, the vector K specifies a spatial distribution for the robots. As the robots are moving into a natural goal formation under



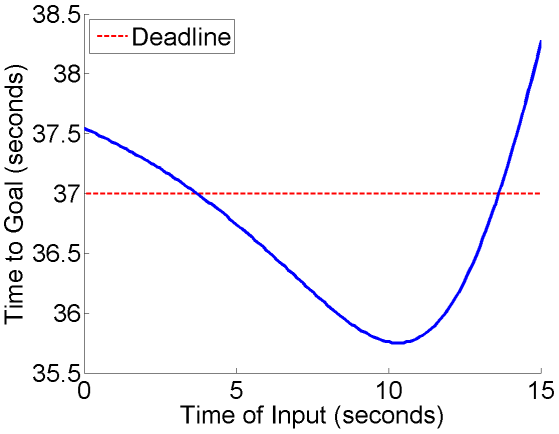
(a) Initial Positions of the Robots



(b) Robot Formation Achieved when No Human Input is Applied

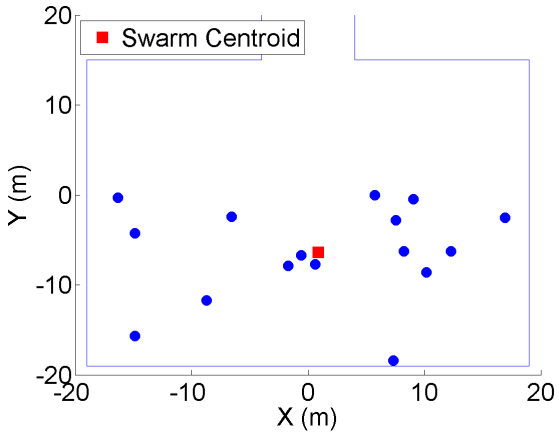


(c) Robot Formation Desired by the Human and Generated when Human Input is Applied

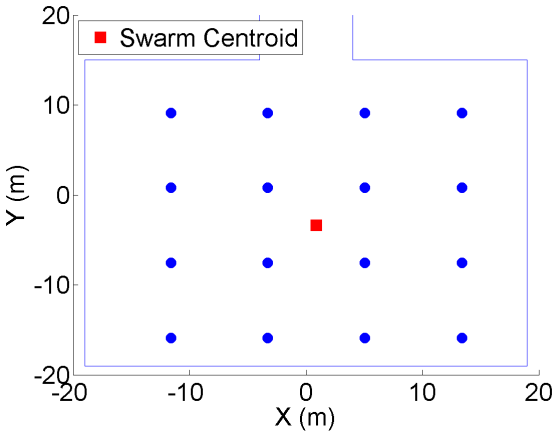


(d) Time at which the Desired Formation is Achieved vs. Time at which Human Input is Applied

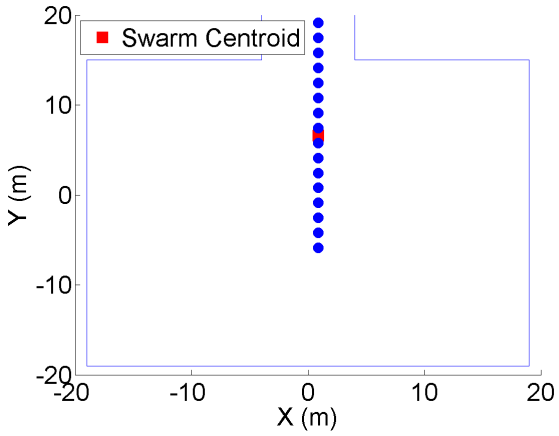
Figure 2.2: Neglect Benevolence During a Configuration Control Task



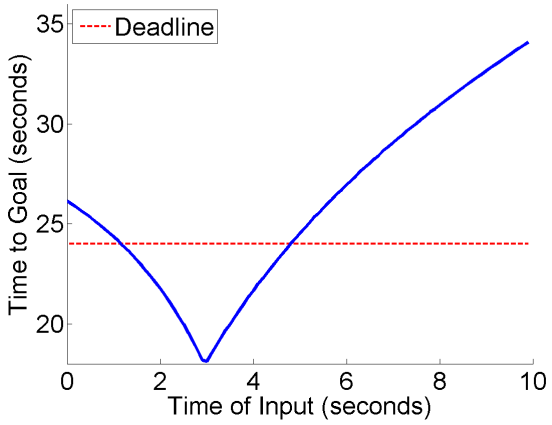
(a) Initial Positions of the Robots



(b) Robot Formation Achieved when No Human Input is Applied



(c) Robot Formation Desired by the Human and Generated when Human Input is Applied



(d) Time at which the Desired Formation is Achieved vs. Time at which Human Input is Applied

Figure 2.3: Neglect Benevolence when Moving Robots into a Narrow Passage. The dotted red line is the deadline within which the human goal must be achieved.

the influence of the system dynamics, the human may become aware of a new goal to be achieved with the swarm prior to a fixed deadline. The human would then want to provide an input to influence the swarm into a different formation required to achieve the new goal prior to the deadline. In this case, the performance criterion comes in two parts: (1) the swarm must achieve the desired formation before the deadline and (2) minimizing the total time required to achieve the goal formation is desirable. To meet the deadline and minimize the time required to achieve the goal, the human may naively try to provide input as soon as possible (i.e. immediately after becoming aware of the new goal). However, the theory of Neglect Benevolence developed in previous sections indicates that there may exist certain states of the system in which applying the input immediately causes the system to have worse performance than delaying the application of input. Specifically, delaying the application of input may increase system performance.

Several simulations demonstrated that this system for generating robot configurations did indeed exhibit Neglect Benevolence. Simulated swarms consisted of between 3 and 25 holonomic robots that could communicate bidirectionally with a fixed subset of other swarm members (i.e. communication graph not assumed to be complete). Each simulated robot could measure only its own position on the map.

Results from one such simulation are shown in Figure 2.2. In Figure 2.2a, the robots are shown initially dispersed across the map. If no human input is provided to the swarm, the system dynamics naturally drive the robots into the circular formation shown in Figure 2.2b. However, when the swarm is in the initial state shown in Figure 2.2a, the human becomes aware of a new goal that requires the swarm to be in the torus formation shown in Figure 2.2c. The human also becomes aware that the swarm must achieve this formation within 37 seconds and that it is preferable to achieve the formation as early as possible. As shown in Figure 2.2d, if the human naively tells the swarm to form a torus immediately at $t = 0$, the swarm does not achieve the formation within the minimum amount of time. In fact, the swarm does not achieve the torus formation prior to the deadline at all, much less in the minimum amount of time! The human must wait approximately 4 seconds before influencing the swarm in order for the swarm to create a torus formation before the deadline. If the human delays the application of input by approximately 10 seconds, then the swarm achieves the torus formation within the minimum amount of time.

Figure 2.3 presents simulation results from another scenario that also requires the swarm to achieve different formations and exhibits Neglect Benevolence. In this simulation, the swarm begins in a room in the initial configuration shown in Figure 2.3a. It is assumed that prior to the start of the simulation the human operator's primary goal was to maximize exploration of the room and the human chose to do so by informing the swarm to form a grid. If no further human

influence is provided, the swarm will naturally move into the grid formation shown in Figure 2.3b. However, immediately after the human operator has applied the input for forming the grid configuration, they are given a new goal that requires moving the swarm into the narrow passage. To achieve this goal, the human informs the swarm to move toward the passage. This input appears as a constant bias term in the system dynamics equation and causes the swarm centroid to move towards the passage. To avoid colling with the walls of the room before reaching the passage, the human must also inform the swarm to create the line formation shown in Figure 2.3c. The first robots in the swarm will reach the passage (or collide into the walls) at 24 seconds, but this implicit deadline is not necessarily known to the human. In the previous section, it was shown that the system dynamics will always ensure the formation is centered on the (in this case, moving) centroid of the swarm. As shown in Figure 2.3d, if the human applies the input immediately, the swarm will not achieve the formation before reaching the narrow passage. There is a narrow window of opportunity between approximately 1 and 5 seconds when applying the input enables the swarm to achieve the new formation prior to reaching the passage. In addition, there is an optimal input time at 3 seconds, when applying the input will enable the swarm to move into the line formation by 18 seconds, well before the 24 second implicit deadline.

2.6 Conclusion

Robotic swarms that autonomously coordinate via simple local control laws are becoming increasingly interesting for applications such as reconnaissance, surveillance and disaster response. These applications are characterized by uncertainty and potential changes of the environment and mission goals. This results in the human redirecting the swarm by changing its goals and behaviors. In prior experimental work in Human Swarm Interaction [79], we observed that system performance often depended on the timing of the operator input to the swarm. In particular, in many instances, system performance improved if the operator, instead of giving input to the system immediately as soon as mission goals changed, delayed giving the new input to the system. We labeled this notion of beneficial delay as Neglect Benevolence. In this chapter, we formalized Neglect Benevolence and presented ways to quantify this notion as well as procedures to compute the optimal time for the operator to give input. We also proved the existence of Neglect Benevolence for a class of linear systems. Experimental simulation results showed that in many situations where the system must reach a specified human goal within some deadline, if the human does not consider the effects of Neglect Benevolence in timing her input, the system may never achieve the goal. This chapter is the first to formally introduce the notion of Neglect Benevolence and present initial results. There remain many additional challenges and areas of

research that we will consider in future work. Some of the issues we plan to address are: (a) obtain further formal results for a sequence of goals and new human inputs, (b) prove the existence of Neglect Benevolence for different measures of system performance (e.g. energy minimization), (c) provide characterization of Neglect Benevolence and optimal timing of human input for non-linear systems, (d) investigate system sensitivity to the phenomenon of Neglect Benevolence with regard to noise in sensor measurements or state estimates.

Chapter 3

Human Approximation of Optimal Input Times

A robotic swarm is a multi-robot system where individual swarm members use control laws based on local spatial information about the environment and/or other members of the swarm within their spatial neighborhood to achieve various emergent collective behaviors, such as foraging or rendezvous. The key advantage of swarms is that the behaviors generated are robust to individual robot failures. Applications of swarm robotics are envisioned in environmental exploration, large scale emergency response and search and rescue and environmental cleanup (e.g. cleaning oil spills). Such missions may require use of a sequence of different swarm behaviors that may need to be imparted to the swarm by a human operator.

Since the state of a swarm not yet at equilibrium is continuously evolving, the supervisor's task is to supply inputs that divert the swarm from its current trajectory to a new desired trajectory. This Trajectory_1-Input-Trajectory_2 sequence therefore defines the basic task that must be addressed in the study of human-swarm interaction (HSI).

There are two related but distinct capabilities needed to supervise a robotic swarm. The first is *comprehension of the swarm's state* and the second is *prediction of the effects of human inputs on the swarm's behavior*. Comprehension of the swarm's state requires a human to perceptually extract relations and regularities in behavior from observable data such as robots' positions and velocities. This can be a challenging task since crucial aspects of the robots' internal states, such as distances from active goals or the conditions controlling current behaviors, may not be directly observable. In [53], for example, researchers were moved to install LEDs on their robots in order to display their internal states to aid in debugging swarm behaviors. Fortunately, regularities in relations and dynamics of widely employed swarm algorithms such as flocking are

readily perceived by human viewers. In fact, the original purpose of the flocking algorithm [69] was to generate computer graphics that humans would find similar to the behavior of flocks of birds or schools of fish.

The second capability, predicting effects of human inputs, requires the operator to develop an "internal model" of the swarm's dynamics. This capability is related to the first, since, if it is possible for the human to discern continuity in the behavior and state evolution of the swarm, then the human would have a better idea of the system dynamics, would be able to predict to some extent what the swarm would do next, and hence be able to choose when to insert input to bring about desired behavior. Even for relatively intuitive relationships such as commanding a leader who is followed by a swarm employing a flocking algorithm, issues involving limitations in communications, robot speed or lags in response require the operator to develop a model of the swarm dynamics in order to effectively control the swarm. For other inputs, such as a command to switch between algorithms, delivered by broadcast or propagation, the relation between input and a desired effect on the swarm may be even less clear. Difficulties in developing internal models of dynamics have been found to be particularly acute when timing of inputs or lags in response are present [79] as is likely in controlling a swarm. A close counterpart lies in industrial processes where substantial lags between input and response have been abstracted to laboratory tasks such as Crossman's water bath [38]. In the waterbath task, operators were asked to control a heater under a beaker to heat the water in the beaker to maintain a setpoint with a lag introduced by putting the thermometer inside a testtube. Later, computerized versions of process control tasks [55] have used multiple tanks, valves and pumps but preserved the lagged integrative response of the waterbath. Common findings in these experiments are that performance improves with practice, but instruction in principles underlying the system does not improve performance [56]. Negative correlations often found between performance and verbalizable knowledge [15] further suggest learning of dynamics may be largely implicit making it difficult to improve or correct human performance.

In view of these results, enabling humans to effectively exert supervisory control on swarms by understanding the underlying dynamics and giving control inputs at the right time to optimize performance objectives presents a huge challenge. In this chapter, we study this challenge and explore ways to better understand whether and how humans comprehend swarm dynamics and also how to best aid them. In particular we want to answer the following questions: (a) Can humans develop a model of the dynamics expressed in moving from one swarm configuration to another, so that they approximate the optimal time to give their input to the swarm? (b) Can we improve human performance by providing an aid which makes the robots' goals visible to the operator and provides a holistic guide to the swarm's state? (c) More generally, can we extend human monitoring and control to distributed systems that do not conform to perceptual principles

by engineering displays for humans rather than requiring swarms to follow inefficient algorithms to make their behavior humanly intelligible?

We conducted a study to investigate human performance at the Neglect Benevolence shape-changing HSI reference task to determine the degree to which human operators could approximate optimal performance, given experiences interacting with system dynamics of a swarm. We ran experiments in two conditions, one where the participants interacted via a usual display (unaided condition) and another where the participants had the use of an augmented display (aided condition). Our results showed that the participants indeed learned over time to improve their performance. Moreover, the success of the participants in the aided condition lends some support to our hypothesis that making swarm consensus variables perceptible to humans would increase the efficiency of human swarm interaction.

This chapter makes the following contributions. First, it presents a framework characterizing Human-Swarm Interaction (HSI) as a sequence of human inputs diverting a swarm's trajectory through state space that is reducible to a series of Trajectory_1-Input-Trajectory_2 events and not restricted to behaviors involving only translation such as flocking, rendezvous, or deploy. Second, it introduces the swarm configuration shape-changing Neglect Benevolence Task as an HSI reference task allowing comparison between human and optimal input timing performance in control of swarms. Third, it introduces a Gestalt-based approach to characterizing the intelligibility of swarm behavior based on the perceptual saliency of the object of consensus. Fourth, it provides an initial test of human performance on the Neglect Benevolence reference task and a test of display augmentation based on characterization of intelligibility.

The chapter is organized as follows. Section 3.1 presents related work. Section 3.2 briefly outlines the notion of Neglect Benevolence and optimal input time. Section 3.3 describes the configuration control and shape-changing task as a HSI reference task allowing comparison between human and optimal performance. Section 3.4 introduces an approach to characterizing the intelligibility of swarm behavior based on the Gestalt principle of common fate and evaluates common swarm algorithms including shape-changing. Sections 3.5 and 3.6 describe the experiment and results comparing human performance with conventional and aided displays at the Neglect Benevolence reference task. Section 3.7 presents discussion and Section 3.8 conclusions and future work.

3.1 Related Work

Robotic swarm systems where a group of robots need to act through local interactions to collectively achieve a variety of behaviors have been extensively studied. Swarms can achieve behaviors such as flocking [14, 24, 69, 74], rendezvous [16], deployment [20], and foraging [30]. Formation control of robotic swarms has been proposed to adjust the spatial configuration of robotic systems so that they satisfy the requirements of deployment tasks or they can adapt to different environments while performing tasks [5, 31, 77]. However, schemes which include a human operator in the loop together with robotic swarms are required for some complicated tasks (e.g. complex surveillance and reconnaissance). For using swarm robotic systems in human-supervised missions, human swarm interaction (HSI) has been studied in [13, 22, 34, 42, 45] primarily at variants of foraging tasks. Study of human involvement in formation control includes transitions between flock and torus [13], leader choice and network configuration [29].

The concept of Neglect Tolerance has been introduced to capture the idea that a human operator can neglect robots that work independently of one another in a multi-robot system for a certain time before system performance degrades [65, 75]. One key issue in Neglect Tolerance is how to schedule the attention of the human operator among the multiple and independent robots so that the neglect time between servicing robots is minimized [52, 54]. In Neglect Tolerance, it is assumed that the robot system's performance will degrade over time and the system will need human interaction to restore performance to a desirable level. In contrast, the concept of Neglect Benevolence in human swarm interaction captures the idea that it may be beneficial for the human operator to wait for a certain length of time before applying input to the system while the swarm state evolves to stabilization (since the swarm performance may not degrade monotonically with time) [57].

3.2 Neglect Benevolence

In a previous experimental study using a foraging scenario [79], it was found that the performance of the human-swarm system was strongly affected by the time between two commands that the human applied to the robots. In particular, it was found that one group of subjects who performed well waited for some time after they issued a command before issuing another corrective command (when they wanted to change the direction in which the swarm was heading). The phenomenon was called *Neglect Benevolence*, since neglecting the swarm for some amount of time led to better performance of the system.

Further analysis of the results [79] found that when the swarm was in a transient state (i.e. moving towards one goal), applying another input (that changes the goal) could have different effects depending on the timing of the inputs.

To determine whether this was a spurious effect or not, [57] reported various simulations of swarm systems, starting at different configurations and performing rendezvous where the operator inputs changed the rendezvous point. The robots were moving under simple control laws given in [36] with a repulsive potential to avoid inter-robot collision and an attractive potential field to maintain swarm cohesion. It was observed that the simulation gave a variety of resulting outcomes depending on the times when the human input was given following the desired change in the swarm goal. More concretely, it was observed that giving the input immediately after the need arose to change the rendezvous point resulted in detaching several robots who subsequently never made rendezvous while some of the delayed inputs led to the whole swarm staying together and completing the rendezvous at the desired point.

Here, we briefly discuss some results shown in [57] that serve as a useful background for the computations and task proposed in this chapter. First, it was shown that Neglect Benevolence is present not only for nonlinear systems (e.g. swarms with collision avoidance) but also for linear systems. Second, a formal definition of Neglect Benevolence was introduced that allows performance to be defined by a variety of measures such as the time or energy used by the swarm to reach the human goal. Third, it was proven that any exponentially stable linear system exhibits Neglect Benevolence. Fourth, an algorithm that computes the optimal input time for insertion of human input to achieve least time to human goal for linear time invariant (LTI) systems was given. Fifth, it was shown that the dynamics matrix of a swarm that performs consensus for controlling swarm configuration control and shape-changing (the reference HSI task proposed in the current chapter) falls within this class of (LTI) systems.

From the results reported in [57], it can be concluded that Neglect Benevolence is a nuanced and quantifiable notion that captures precise calculation of "too early" or "too late" in inserting the human input. In other words, if a system is neglected for a very long time, it may take a very long time to converge. On the other hand, if a system is neglected for too little time it may also take a very long time to converge. Therefore, in order to guarantee timely convergence, a bounded Neglect Benevolence must be determined, so that the neglect interval is neither too small nor too large. Hence, another way of looking at the study in this chapter is to determine whether humans have the ability to gauge the bounds of Neglect Benevolence.

We consider configuration control and swarm shape-changing as a base task within which to study Neglect Benevolence and optimal input timing in human swarm interaction. In the next section, we present more details about the underlying formal model and dynamics of this task.

3.3 Configuration Control as a Base Human-Swarm Interaction Task

In this section, we describe the dynamics of a swarm system using consensus for configuration control. Individual robots within the swarm communicate with their neighbors. The communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ captures the connectivity of robots within the swarm. An individual robot is represented as a vertex $v_i \in \mathcal{V}$ of the graph and the edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ represent communication links such that each edge $(v_i, v_j) \in \mathcal{E}$ indicates that robot $v_i \in \mathcal{V}$ can communicate with robot $v_j \in \mathcal{V}$. Assume that all communication links are bidirectional. This means the communication graph is undirected (i.e. $(v_i, v_j) \in \mathcal{E} \Rightarrow (v_j, v_i) \in \mathcal{E}$). The adjacency matrix $\mathbf{A} = [a_{ij}]$ is a mathematical representation of the graph where where element $a_{ii} = 0$, $a_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$, and $a_{ij} = 0$ if $(v_i, v_j) \notin \mathcal{E}$. The Laplacian matrix $\mathbf{L} = [l_{ij}]$ can be obtained from the adjacency matrix with each element $l_{ii} = \sum_j a_{ij}$ and $l_{ij} = -a_{ij}$. Since the graph is undirected, the adjacency matrix and the Laplacian matrix are both symmetric. When the graph is connected, the Laplacian matrix is singular and has one eigenvalue equal to 0 and a corresponding eigenvector $\mathbf{1}$. Note that $\mathbf{1}$ represents a column vector with all entries equal to 1.

The joint state of the robotic swarm is given by the state vector $\mathbf{x}$, which is composed of the concatenated states x_i of individual robots within the swarm. In continuous-time averaging consensus, the change in the state $\dot{x}_i$ over time t of an individual robot v_i is based on averaging the states of neighboring robots. Two robots v_i and v_j are called neighbors if they can communicate with each other (i.e. $(v_i, v_j) \in \mathcal{E}$). As presented in [68], the evolution of the joint state of a system performing averaging consensus can be described by the following equation, where $\mathbf{L}$ is the Laplacian matrix of the system's communication graph.

$$\dot{\mathbf{x}}(t) = -\mathbf{L}\mathbf{x}(t) \tag{3.1}$$

Note that Equation (3.1) describes a stable linear time-invariant (LTI) system. The Laplacian matrix $\mathbf{L}$ has all non-negative real eigenvalues, so $-\mathbf{L}$ has only non-positive real eigenvalues. Since all eigenvalues of the system are real and non-positive, the system with dynamics described by Equation (3.1) will always move monotonically from any initial state $\mathbf{x}_0 = \mathbf{x}(0)$ towards a final state $\mathbf{x}_f = \lim_{t\to\infty} \mathbf{x}(t)$ that lies within the null space of the system [57]. For a swarm with a connected communication graph, the final state is any state in which the robots all agree with each other. That is, $\mathbf{x}_f = c\mathbf{1}$ where $c \in \mathbb{R}$.

When the robot states are their spatial positions, the averaging consensus dynamics in Equation (3.1) result in rendezvous at the centroid of the robots' initial positions. In [57], this algo-

rithm is extended to incorporate an input vector K.

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

It was shown in [57] that these dynamics cause the system to evolve towards a final state $\mathbf{x}_f$ that (a) preserves the centroid of the initial positions of the robots and (b) preserves the differences in corresponding entries of the input vector K. Thus, if K specifies the robot positions for a formation in some arbitrary but consistent coordinate frame, a robot swarm with the dynamics in Equation (3.2) will generate the specified configuration around the centroid of the initial positions. Thus, a robot swarm could be performing simple consensus and a human operator could broadcast an input vector K and with only local interactions, the robots would still generate the desired configuration.

This introduces the important question of when a human operator should apply a particular input. Specifically, if the human operator has already specified an input K_{old} that generates a particular configuration, then when should the operator specify a new input K_{new} to generate a new configuration (i.e. change the goal of the swarm) as quickly as possible? Human intuition would indicate that the operator should specify K_{new} as soon as possible if they want to obtain the new configuration as quickly as possible. However, as shown in [57], this intuition is provably wrong in the general case! For any LTI system, given an input K, it is always possible to find a state $\mathbf{x}$ in which it is beneficial to delay the application of the input. Since this is non-intuitive to the human operator, this motivates the need for appropriately aiding the operator in finding or approximating the optimal time to apply the input, maximizing the performance of the robot swarm.

3.4 Intelligibility of Swarm Behavior

The emergent behavior of a swarm results from many local interactions among its members. Most commonly, some variant of consensus (averaging values among neighbors) is used for the coordination giving rise to swarm behaviors. Perceiving an emergent swarm behavior requires the observer to recognize and associate similarities among the many local interactions among robots. Consequently, human perceptual access to swarm behavior depends on these local interactions/agreement being expressed in a way in which salient similarities and relations (consensus) can be recognized. Recognition of these regularities and the swarm's trajectory is a prerequisite for monitoring and decision making. The extraordinary ability of humans to perceive biological motion has been well established since Johansson's [40] experiments in which human actors with point-lights attached to major joints were recorded and later viewed as images

showing only a few moving dots. Viewers easily identified the human figures and their activities, such as walking, even in the presence of substantial "masking" by other dots [28]. Even hidden characteristics of motion such as the weight of a box [70] or distance of a thrown object [43] were discernible to viewers of these point-light displays. Inverting the imagery (upside down human walkers), however, eliminates almost all of the effect [10] leaving observers unable to discriminate between scenes containing a walker and those with only a "mask". These results suggest a strong ecological and/or learned component to human abilities to perceive biological motion that may not fully extend to behaviors of a swarm. This suggests that the more general Gestalt principles of *proximity*, *similarity*, *closure*, *symmetry*, *continuity*, *common fate*, *pragnanz* and *past experience* may provide the best guidance for identifying representations likely or unlikely to lead to the perception of swarm behaviors. Of these principles, only *common fate* applies specifically to motion, while the rest largely prescribe how parts come to be perceived as wholes and could come into play as robots approach well defined formations.

The most common display of a swarm comprises spatially related icons on a plane or in a 3D space corresponding to robot positions. Swarm behavior following standard algorithms such as *flocking* or those collected in [16] such as *rendezvous*, *deploy* or *boundary following* is displayed through movements of these icons. Fortunately, these displays (moving dots perceived as patterns) closely resemble those studied by psychologists and vision researchers so there is already research into the factors supporting the perception of group behavior in this context. *Common fate* refers to the grouping factor that leads a previously invisible grouping of dots to pop out of a noisy background when moved and has been found to be the most important factor in perceiving coordinated behavior although *proximity* [78], *similarity* [26] and *pragnanz* in the form of grouping features such as collinearity [78] have all been shown to play important roles and to interact [78] in strengthening the effects of *common fate*. Humans are very sensitive to perceiving *common fate* groupings with success at signal-to-noise ratios as low as 1-2% [76] and the effect is relatively robust to divergence among trajectories as long as some coordinated global direction is maintained [82]. There are suggestions, however, that *common fate* may be limited to a single group at a time [48]. For example, of two overlapping groups only one would appear to move coherently. Effects have been found strongest for linear or circular trajectories [83]. While spatial characteristics predominate in perception of moving groups other features such as common color can also contribute [26].

By this analysis, even without the benefits of biological motion, flocking behavior in which agents adopt the average heading and velocities of their neighbors seems ideally suited for perceptual recognition. In either its linear or torus form [13], the interaction among agents is precisely that giving rise to *common fate* while the regulation of distance (*proximity* and *pragnanz*) enhances this effect. For the rendezvous and deploy algorithms, the consensus process should

be less strongly conveyed because while the global movement inward or outward and reflected in local interactions should be perceivable [83] it lacks the matching of velocities and headings, which are the dominant contributors to the perception of *common fate* [78].

By a similar analysis we would predict that the behavior of a swarm following the shape-changing algorithm would be extremely difficult for a human observer to follow because in moving from one configuration to another neither headings nor velocities are matched (*common fate*) nor constant distances nor boundaries maintained (*proximity* and *pragnanz*). That is to say that the movement of the swarm toward consensus as perceptually manifested by a centroid is masked by the formation. Since the consensus process is not perceptually accessible from robot behaviors, the human monitor would be left with the impression of robots that are moving about independently and be unable to either monitor their progress toward a goal or compose an input to alter that progress. As the target formation is approached, however, the entire set of Gestalt principles governing perception of form (*proximity*, *similarity*, *closure*, *symmetry*, *continuity*) would come into play making the goal and progress toward it increasingly apparent in this second phase.

These analyses suggest that intelligibility of swarm behavior could be improved for algorithms in which progress toward consensus is not perceptually available by augmenting the display in a way that makes this information available. Since the perceptually uncoordinated movements of robots to their positions in the target formation cannot be eliminated, the display can only be augmented in ways that make the robots' goals and progress toward their goals more apparent. A potential augmentation of this sort would be to draw lines from the robots' current positions to their predicted locations in the new formation. As the swarm evolves toward its initial goal shape these lines would shorten and lengthen until the supervisor chooses to divert the swarm to the desired shape at the point he/she judges to be the closest (shortest lines). In this augmentation, the line between each robot's current position and its final position is a visual proxy for the distance in the swarm's state space between current and goal state. The swarm's state space has very high dimensionality and cannot be easily visualized, so this proxy is used as an aid. The next section describes an experiment comparing a display augmented in this way with a conventional display.

3.5 Experimental Design

Our initial experiment investigating human performance at the Neglect Benevolence HSI reference task was to determine the degree to which human operators could approximate optimal performance given experience interacting with system dynamics (implicit learning). Comparison of human and optimal performance is a standard practice in the study of human-machine systems

for benchmarking human performance and evaluating human competence within a work domain. While we would expect to assign easily automatable tasks such as input timing to the system, it may be important to allow the human to intervene in response to "out of band" information or other anomalies. In addition, the possibility of manual control may be an important element for preventing out-of-the-loop effects or complacency and enabling the operator to effectively monitor the system for failures or abnormal behavior. Possible outcomes included almost optimal performance, biased performance favoring late inputs ("finding the bottom" heuristic), idiosyncratic performance indicated by a wide dispersion between early and late inputs or some other explainable outcome. These are descriptive questions answerable by descriptive statistics.

We used a mixed-model design for our experiment, which divided subjects into two groups and had one group interact with the swarm through a conventional display showing only robot positions (unaided condition), while the other group interacted with the swarm through a display that included a Gestalt-based augmentation to the visualization of the swarm (aided condition). Interactions with the swarm were divided into discrete trials. Each group interacted with the swarm in the same number of training and test trials. The swarm dynamics and initial state remained consistent between corresponding trials in the unaided and aided condition. Thus, for the mixed-model design, the within-subject variable was the trial number and the between-subject variable was the presence or lack of the Gestalt-based visual aid. A regression against trials was used to measure practice effects. The comparison between the display with the Gestalt-based augmentation and the conventional display provides an initial test of our hypothesis that swarm behaviors can be made more intelligible and controllable by making progress toward consensus perceptually assessable.

3.5.1 Method

Task

The experimental task involves human interaction with a simulated robotic swarm performing consensus on robot positions. The swarm includes the capacity to incorporate an input as described in Section 3.3. The task is for the subject to choose the best time at which to provide an input diverting the robotic swarm from the shape toward which it is initially moving (equilibrium point of the dynamical system) to another desired shape (new equilibrium point). This matches our earlier depiction of the canonical HSI task as redirecting the swarm — modelled as a dynamical system — from one "natural goal" state (Formation 1) to another "desired goal" state (Formation 2). If the system in a Neglect Benevolent state, it is beneficial to delay the application of the input to minimize the total time required to converge to the desired formation. Figure 3.1

shows an example of how delaying the application of the input could reduce the time required to reach the goal state.

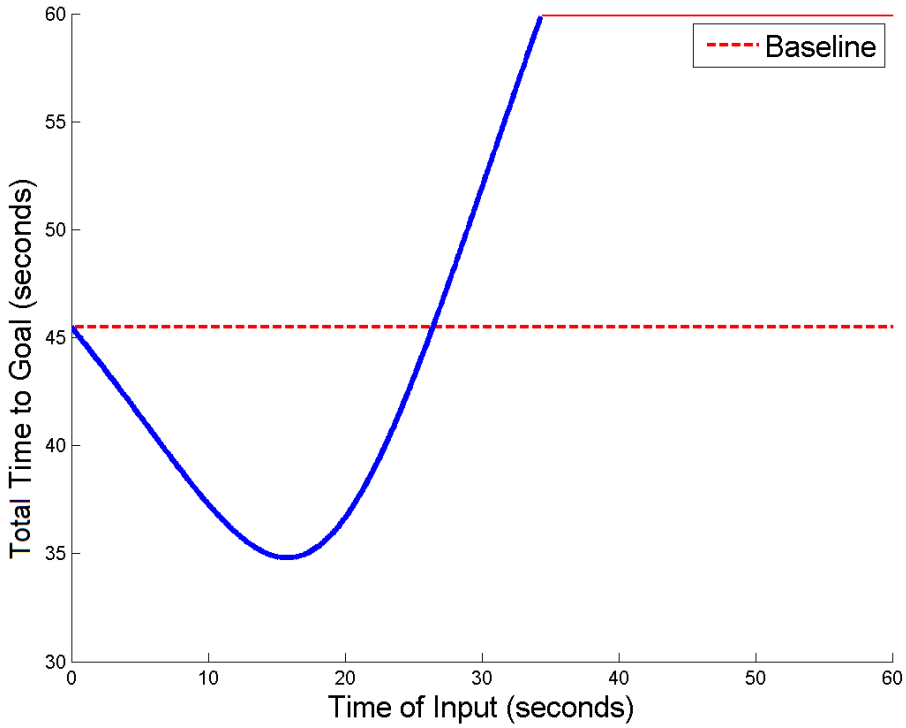


Figure 3.1: This plot shows the relationship between convergence time (i.e. total time to goal) and input time for one of the trials in our experiment. The convergence time if the input is applied immediately at the start of the simulation is shown as the baseline time. Each trial has a maximum length of 60 seconds. For this trial, if the input is applied after approximately 33 seconds the swarm would not converge within the 60 second limit.

Participants first read the standard instructions on how to use the interface to interact with the swarm robots and then proceed to complete each trial in the experiment. In the aided condition, there is a line between each robot's current position and its final position in the desired formation. These lines help participants visualize the distance between the robots' current positions and the desired formation (Formation 2). Since the swarm dynamics are LTI as described in Section 3.3, when the collective distance represented by all the lines is larger, more time is required to reach the desired formation. Formation 1 and Formation 2 are shown in thumbnails at the top of the screen following a summary of instructions.

The experiment has 12 training trials to familiarize participants with the system and allow them to observe the asymmetric effects of leading/lagging the optimal input time. As learning of system dynamics is presumed implicit, familiarization is needed to engender common strategies so performance might be compared. During the training trials, participants are informed of the

effects of applying the input too early/late and compare them with the optimal time. The training trials are divided into two types: *passive*, where participants only observe, and *active*, where participants give input themselves. The first 6 trials are *passive*, where participants observe three simulation panels situated below thumbnails depicting Formation 1 and Formation 2. All three simulations begin with the robots in the same random initial configuration. When a user clicks "Begin", the swarms in all three panels begin moving toward Formation 1 until given an input (at a different time for each panel). Visually, upon receiving the input, the swarm robots change color from red to green. Each panel shows how the input redirects the configuration of the swarm towards Formation 2. Upon converging to Formation 2, the convergence time is shown. The left, middle and right panels each display the effects when the input is applied "too early", at the optimal time, and "too late" respectively (see Figure 3.2). The next 6 trials are *active*, where participants have the chance to apply the input themselves and get feedback as to whether they gave the input too early, too late, or optimally (see Figure 3.3). Participants are again presented with thumbnails showing Formation 1 and Formation 2, but followed by only one simulation panel. The panel shows the robot swarm in a random initial configuration. After the participant clicks "Begin", the swarm starts moving toward Formation 1. The participant then has the chance to give an input by clicking "Activate New Formation" at any time before the end of the simulation. Once the swarm has converged (or the simulation has reached the 60 second time limit), the convergence time resulting from the input and feedback as to whether their input was too early or too late (see Figure 3.3) is displayed.

After the subjects finish the 12 training trials, they are presented with 36 test trials in the same condition (aided or unaided). In the test trials, the subjects are presented with the same screen arrangement (see Figure 3.4) as in the *active* training trials, but although they are able to observe the convergence time, they are not given feedback as to whether their input was too early or too late. The participants are instructed to impose the new goal on the swarm at the time leading to the earliest convergence.

Participants and Data Collection

A total of 44 participants were recruited (22 for aided condition, 22 for unaided condition) from the University of Pittsburgh and Carnegie Mellon University communities through online solicitation. Data was collected through a HTML5 / JavaScript web application that recorded each subject's trial data anonymously in a secured MySQL database. Participants were randomly assigned to either the control group (unaided condition) that interacted with the swarm through a conventional interface or the experimental group that interacted with the swarm through an interface with a Gestalt-based augmentation (aided condition). Participant input times were recorded

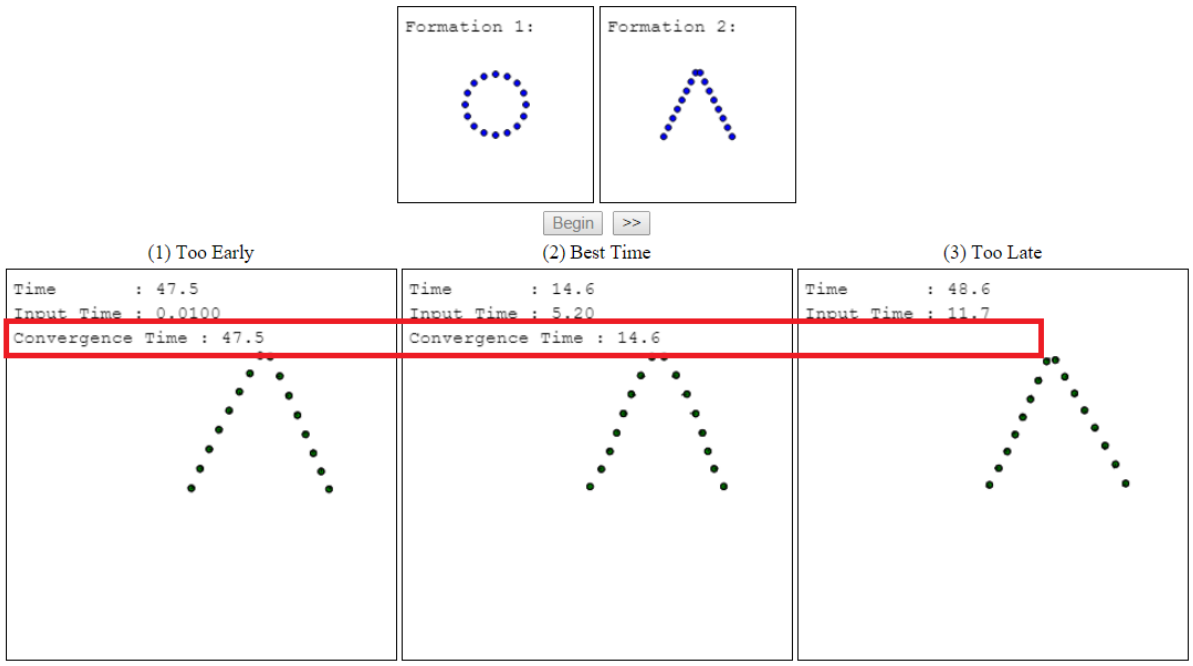


Figure 3.2: For each trial in the first (passive) training phase, participants were shown three simulations. In all simulations, robots started in the same initial positions, but the input telling them to move to Formation 2 was applied at different times: (1) too early, (2) at the best time, (3) too late. After the swarm converged to the final formation (see panel 1 and 2), participants were shown the time taken for the robots to converge.

for each trial. Optimal input times for each trial were computed offline using the algorithm in [57].

3.6 Results

Input time deviation and convergence time served as dependent variables. Input time deviation ($|t_{\text{input}} - t_{\text{optimal}}|$) was defined as the absolute difference between the time chosen by participants to divert the swarm and the optimal diversion time. Convergence time was defined as the interval from the beginning of the trial until the swarm converged to the desired formation (Formation 2). Input time measured the participants' deviation from optimality directly while convergence time measured the effect of these suboptimal inputs on performance. Since delays that were too long could lead to much longer convergence times than those that were too short, strategies favoring earlier inputs might be a rational response for participants having difficulty in approximating

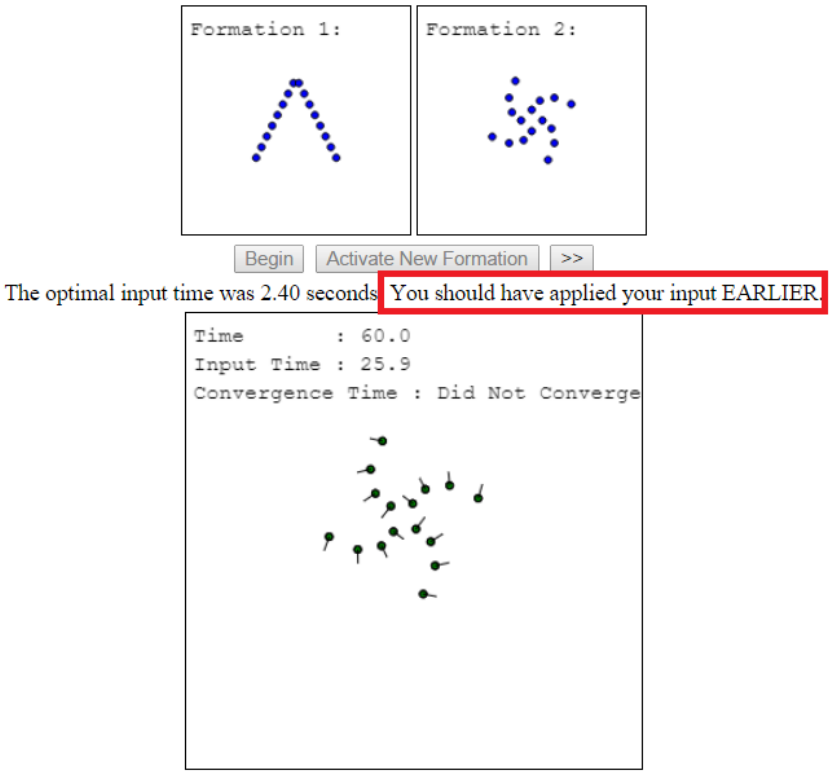


Figure 3.3: For each trial in the second (active) training phase, participants were asked to apply the input at whatever time they thought would minimize the total time required for the robots to converge to Formation 2. At the end of the trial, participants were told the optimal input time and whether they should have applied their input earlier or later.

the optimal interval making convergence and input times slightly different measures. Independent variables were Display type (conventional or aided), Transition Type (circle-spiral, circle-V, spiral-circle, spiral-V, V-circle, V-spiral), and trial number.

Data were analyzed both in an ANOVA design treating Display type and Transition type as factors and a regression incorporating trial numbers as well. An ANOVA found main effects for both Display types (F1,1572= 6.214, p=.013) and Transition type (F1,1572=33.402, p¡.001) for input time deviation. An ANOVA for convergence time also found main effects for both Display type (F1,1572= 3.387, p = .006) and Transition type F1,1572=45.418, p¡.001). Both groups on average preceded the optimal time with participants in the aided display condition responding nearly twice as early (M=-1.23 sec, SD=5.11) as those in the unaided condition (M=-.65 sec, SD=7.49) although this difference was not significant at the p¡.05 level. A Mann-Whitney U test of deviations from optimal input times, however, found that participants in the aided condition (mean rank=767.63) were significantly closer to the optimum (z=-2.164, p=.03) than those in the

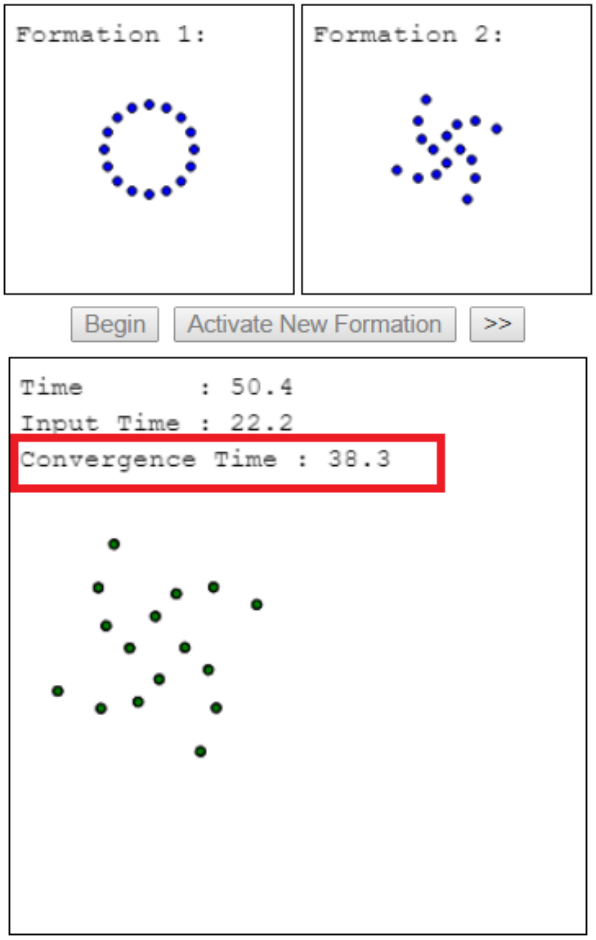


Figure 3.4: For each trial in the test phase, participants were asked to apply the input at whatever time they thought would minimize the total time required for the robots to converge to Formation 2. They were not given any feedback.

unaided condition (mean rank=817.37). A comparison of deviations from optimal convergence times found that participants in the aided condition (mean rank=762.55) were significantly closer to the optimum than those in the unaided condition (mean rank=822.45); z=-1.97, p=.009 on this measure as well. Regression of input time deviations against trials found a significant improvement ($F_{1,34}$=4.918, p=.033) in the unaided Display condition but not in the aided condition.

3.7 Discussion

Our results as shown in Figure 3.5 indicate that humans did, in fact, learn to approximate the optimal input time over the course of the experiment. This suggests that both Display types

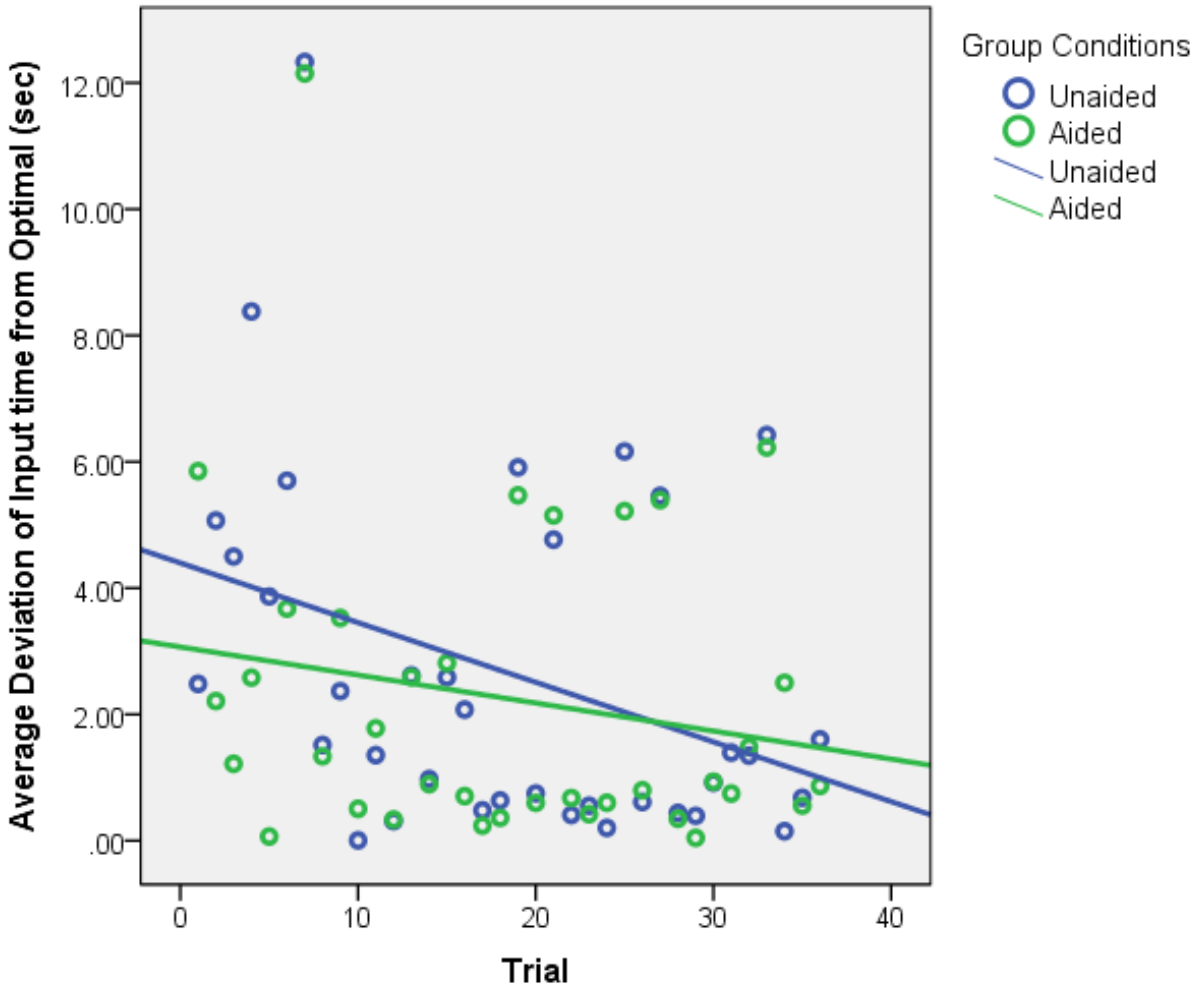


Figure 3.5: This plot shows the average absolute deviation of the participants' input time from the optimal input time for each trial in the unaided and aided condition.

provided sufficient cues for humans to learn to time their input in order to decrease convergence times. The quality of this approximation, however, began at a high level for the group using the aided Display but developed only slowly for those in the unaided condition. We believe this occurred because the lines drawn to terminal locations in Formation 2 on the augmented Display provided viewers with a reference for judging the swarm's evolution while the movement of robots in the unaided Display did not. Relying only on Gestalt influences over the perception of form as formations were approached, unaided participants did not have a basis for earlier responses. This conjecture is supported by the finding that those in the aided condition responded on average earlier than those in the unaided condition There are several possible explanations for the improvement in performance of participants in the unaided condition. Participants might

have acquired an implicit model of shape-changing dynamics that allowed them to better assess the swarm's state as it evolved. Alternately, they may have developed greater sensitivity in recognizing an evolving formation's similarity to their goal formation. A third alternative might be that participants learned a timing heuristic for delaying input. These and similar questions can only be answered through additional properly designed experiments to test those explanations.

The success of the augmented condition in improving the performance of naive participants lends some support to our hypothesis that difficulties can arise in interaction with swarm systems because the variables on which the swarm is coordinating are not perceptually accessible to the human. Many distributed algorithms involving human interaction such as flocking, biology-mimicking control through leaders and predators, or physicomimetic direction via potential fields depend in some part on their intelligibility to human users and their programmers. If we could develop principled methods for augmenting the behavior of swarm algorithms with display of variables under consensus it could extend human control and use to a wider range of potentially more efficient algorithms.

3.8 Conclusion

Some of the main contributions of this work are methodological and lie in providing a framework for a more controlled study of human-swarm interaction. Characterizing HSI as the act of diverting a swarm's trajectory through state space from the current goal to an alternate goal provides a new basis for evaluating HSI in isolation from normally confounding factors. In particular, by focusing on progress toward goal states rather than simply measuring transient effects of behavior, it allows an experimenter to define a strict performance reference for human behavior unlike earlier efforts such as [45] which have relied on comparing performance to that of intelligent agents.

The shape-changing Neglect Benevolence Task provides an example of an HSI reference task of this sort that allows direct comparison between human and optimal performance in control of swarms. The methods and tools developed in this process provide a roadmap for other researchers wishing to develop distributed algorithms and displays that can be evaluated in these ways. We have presented and provided an initial test of a Gestalt-based approach to characterizing the intelligibility of swarm behavior based on the perceptual salience of the object of consensus. While results are preliminary they are promising in providing a closer linkage between human cognition, distributed algorithms and the way we display them.

Chapter 4

A Timing-Only Approach to Adversarial Influence of Swarm Consensus

Multi-agent consensus is essential to the operation of many distributed systems including sensor networks [64], social networks [1] and multi-robot systems such as robotic swarms. When each agent in the system begins with a possibly different value for the same variable of interest, global consensus is said to be achieved when all agents in the system agree upon the same value for the variable. Global consensus may be achieved at a particular time instant or it may be achieved asymptotically as differences between the values assigned by each agent to the variable of interest decrease over time. In sensor networks [64], consensus may be used for estimating a quantity of interest via measurements captured by each of the sensors without having a central processor to perform sensor fusion. In social networks [1], consensus is often used to model the dynamic evolution of opinions held by different members of society. It can be used to study phenomena such as opinion polarization or the effects of disparity in the influence of particular members of society. In distributed multi-robot systems such as robotic swarms [12][46], multi-agent consensus enables distributed coordination using only local information and no central coordinator. These features make robotic swarms that employ consensus-based behaviors [60] robust to failures and scalable since members can be added and removed with minimal system reconfiguration. Swarm behaviors such as formation control and rendezvous are based on linear averaging consensus using agent positions as state variables, while other swarm behaviors such as flocking [69] employ a form of non-linear averaging consensus.

Mathematically, the consensus problem has been studied in a wide variety of settings and can be characterized along multiple dimensions including but not limited to (a) continuous-time or discrete-time updates [50], (b) synchronous or asynchronous updates [33], (c) fixed or time-

varying network topology [63][84], (d) adversarial agents (that can cause byzantine failures) or cooperating agents [71], (e) homogeneous or heterogeneous agents [67]. In this chapter, we study a multi-agent team that asynchronously performs discrete-time averaging consensus. The team consists of cooperating agents that are unaware that some of the agents could be compromised by an adversary. For example, this system could effectively model a robotic swarm of citizen agents that might have been compromised through the introduction of mole agents [72] or a robotic swarm with a subset of agents that have been hacked. The agents on our team have a fixed interaction topology (i.e. each agent has the same set of neighbors over time), but they are heterogeneous in the sense that each agent executes the discrete-time consensus update (i.e. sets its current state to the average of its own and its neighbors' states) with a possibly different frequency. In addition, there is a different delay before the first update for each agent on the team. These delays enable us to model real world features such as initialization delays, timers or even deliberate input delays. It has been shown in the literature that delaying an input to a robotic swarm can actually improve the performance of the swarm [58] — a phenomenon known as Neglect Benevolence — and in some cases it is only possible to meet a deadline by applying a sufficient delay [57].

While unweighted synchronous averaging consensus guarantees that a connected network of agents will always asymptotically converge to an agreement point that is the average of the initial states of each agent [68], there are no such guarantees for general asynchronous consensus. However, in a real-world multi-robot system, asynchrony is is almost inevitable. There has been substantial work in the literature studying how to mitigate the effects of asynchrony to ensure convergence [33] and even specify protocols under which convergence to the same agreement point as synchronous consensus is guaranteed [18]. In contrast to the prior work, in this chapter we investigate how asynchrony in a multi-robot system periodically applying standard consensus updates can actually be exploited to change to agreement point of the team. Particularly, we are interested in situations where we are unable to directly affect the interaction topology or the update rules used by the agents. Instead, we show that by only changing the update periods (equivalently, frequencies) or initial update delays of a subset of agents on the team, we can influence the agents to move towards a desired agreement point of our choosing.

Studying the effects of changing update periods and delays is essential to understanding potential vulnerabilities in distributed robotic systems such as robotic swarms that rely heavily on local coordination through consensus to generate emergent global behaviors. Changing the update period or introducing a delay is often a very simple form of attack since, in the worst case, it only requires access to the communication medium rather than direct access to individual agents (e.g. periodically jam messages to introduce delays or stretch update periods). Understanding these effects can lead to the development of mitigation strategies or tradeoffs for not implementing

such strategies. Studying these effects can also enable the development of strategies to exploit them for beneficial effect (e.g. introduce delay to improve performance without changing the update rules or network topology).

Our contributions are (a) formalizing this problem, (b) proving convergence properties enabling us to reformulate the problem to more efficiently find optimal solutions and (c) illustrative simulations highlighting the utility of our results in robotics and multi-agent applications.

4.1 Related Work

A large variety of different multi-agent consensus problems have been studied in the extant literature. A survey of the basic properties of systems performing linear continuous-time and discrete-time consensus is given in [68]. The problem of convergence of consensus protocols in discrete-time systems with arbitrary time-varying delays is explored in [33]. When the communication graph topology is fixed, it is shown that if the graph contains an directed spanning tree and the delay along any edge in the graph is bounded and at least one agent in the system can access its own state without delay, then the system is guaranteed to asymptotically achieve global consensus. Interestingly, an equivalence is established between a synchronous system with a time-varying topology and an asynchronous system with a fixed topology and zero communication delays. Thus, similar conditions for convergence are established for the former. Our work is distinguished from prior work by exploiting asynchrony for desired effects rather than treating it as something that creates undesirable artifacts for averaging consensus protocols.

It was shown in [57] that delaying an input to dynamical system, such as a robotic swarm performing consensus, can improve the performance of the system (e.g. minimizing the time required achieve an objective). This phenomenon was termed Neglect Benevolence. These results were applied to a robotic swarm performing a form of biased consensus to generate formations around the centroid of the initial positions of the robots using neighbor-only relative measurements. It was shown that sometimes delaying an input is necessary to meet a deadline. In this chapter, we explore the impact of timing on a team performing consensus by not only considering the inclusion of a delay, but also changing the update period.

4.2 Problem Formulation

In this section, we first introduce some preliminary definitions and notions that we use to formalize our problem. We then describe our system model. Finally, we provide a formal mathematical statement of our problem.

4.2.1 Preliminaries

Consider a group of n agents. The state of agent i at time $t \in \mathbb{Z} : t \geq 0$ is given by $\mathbf{x}_i(t) \in \mathbb{R}^m$ and the joint state is given by $\mathbf{x}(t) = \begin{bmatrix} \mathbf{x}_1(t) & \mathbf{x}_2(t) & \dots & \mathbf{x}_n(t) \end{bmatrix}^\top$. Global consensus [68] is said to be achieved asymptotically if $\forall i, j : \lim_{t\to\infty} \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|_2 = 0$. When $m = 1$, this condition can be equivalently written as $\lim_{t\to\infty} \mathbf{x}(t) = c\mathbf{1}$ for some constant $c \in \mathbb{R}$.

A real square matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$ is called right stochastic if the sum of each row is 1 (i.e. $\forall i : \sum_{j=1}^{n} D_{ij} = 1$). The product of two right stochastic matrices is also right stochastic. A right stochastic matrix $\mathbf{D}$ is called regular if there exists n such that all entries of $\mathbf{D}^n$ are positive. A Markov chain [41] with a regular transition matrix is aperiodic and irreducible (i.e. all states communicate), which means $\lim_{n\to\infty} \mathbf{D}^n = \mathbf{1}\mu^\top$ for some stochastic vector $\mu \in \mathbb{R}^n : \mu^\top\mathbf{1} = 1$ called the stationary distribution of the Markov chain. In fact, μ is the left eigenvector of $\mathbf{D}$ corresponding to the eigenvalue 1 (i.e. $\mathbf{D}^\top\mu = \mu$).

4.2.2 System Model

We consider a team of n agents with unique identifiers $\mathcal{I} = \{1, 2, \dots, n\}$ interacting with each other. The initial state of agent i is given by $\mathbf{x}_i(0) \in \mathbb{R}^m$ and joint initial state of the team is given by $\mathbf{x}(0)$. Each agent in the team has a fixed set of neighbors. The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ captures the neighbor information for this team, where the nodes $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ represent agents and each edge $(v_i, v_j) \in \mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$, represents that agent j communicates with agent i (i.e. agent i has access to the state information of agent j). We assume that every node has access to its own state information (i.e. $(v_i, v_i) \in \mathcal{E}$). The adjacency matrix for this graph is given by $\mathbf{A} \in \mathbb{R}^{n\times n}$ and contains non-negative binary elements such that $A_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$ (otherwise, $A_{ij} = 0$). The set of neighbours of node v_i is $\mathcal{N}_i = \{v_j \in \mathcal{V} \mid (v_i, v_j) \in \mathcal{E}\}$. Note that $\forall i : v_i \in \mathcal{N}_i$.

Each agent on our team periodically updates its own state as a function of the states of neighboring agents. The update rule used by all agents on our team is exactly the same: discrete-time averaging consensus. Specifically, on every update, each agent i updates its own state to the

average its own state and the states of neighbor agents $\mathcal{N}_i$. In contrast to many consensus systems studied in the literature, in our system each agent executes the same update rule, but with a possibly different period and phase shift. The update period of each agent is given by $T_i \in \mathbb{Z}$ and the delay prior to executing the first update is given by $\tau_i \in \mathbb{Z}$.

Based on the above system model, the set of agents that execute an asynchronous discrete-time consensus update at time t is given as follows.

$$U\left(t, T_1, \ldots, T_n, \tau_1, \ldots, \tau_n\right) = \left\{j \in \mathcal{I} \mid t - \tau_j \equiv 0 \mod T_j\right\} \tag{4.1}$$

Due to limited space, we will drop the explicit dependence of this set on the update periods and delays and write it as $U\left(t\right)$. Now the state evolution of each agent can be described using the following recurrence relation.

$$\mathbf{x}_i\left(t+1\right) = \begin{cases} \frac{1}{\sum_{j=1}^{n} A_{ij}} \sum_{j=1}^{n} A_{ij} \mathbf{x}_j\left(t\right) & \text{if } i \in U\left(t\right), t \geq \tau_i \\ \mathbf{x}_i\left(t\right) & \text{otherwise} \end{cases} \tag{4.2}$$

where $i, j \in \mathcal{I}$ and $t \geq 0$ is the time.

4.2.3 Problem Statement

Imagine that we can influence a subset of agents on the team given by $\mathcal{C} \subseteq \mathcal{I}$. However, we cannot change the update rule, the initial states or the set of neighbors for each agent. We can only change the update periods $\{T_i \mid \forall i \in \mathcal{C}\}$ and the initial update delays $\{\tau_i \mid \forall i \in \mathcal{C}\}$. As shown in Figure 4.1, changing these periods and delays can significantly change the final agreement point to which the team of agents converge. Consider the situation where we would like the final agreement point of these agents performing consensus to be as close as possible to a desired value $\mathbf{x}_g \in \mathbb{R}^m$. Our problem may then be written formally as follows.

$$\begin{aligned} &\underset{\forall i \in \mathcal{C}: T_i, \tau_i}{\arg\min} && \sum_{k=1}^{n} ||\mathbf{x}_g - \lim_{t \to \infty} \mathbf{x}_k\left(t\right)||_2^2 \\ &\text{subject to} \\ &\forall k : B_k && = \frac{1}{\sum_{j=1}^{n} A_{kj}} \\ &\forall k : \mathbf{x}_k\left(t+1\right) && = \begin{cases} B_k \sum_{j=1}^{n} A_{kj} \mathbf{x}_j\left(t\right) & \text{if } k \in U\left(t\right), t \geq \tau_k \\ \mathbf{x}_i\left(t\right) & \text{otherwise} \end{cases} \\ &\forall i \in \mathcal{C} : && 0 \leq \tau_i \leq \tau_{\text{upper}} \\ &\forall i \in \mathcal{C} : && 1 \leq T_i \leq T_{\text{upper}} \end{aligned} \tag{4.3}$$

4.2.4 Motivating Example

In Figure 4.1, we show a concrete example that illustrates our system model and various aspects of our problem. A team of seven agents has the interaction topology shown in Figure 4.1a. If agent states evolve according to the consensus update rule using their default update periods and initial delays (i.e. no delay prior to first update), their states evolve as shown in Figure 4.1b. However, changing the update period and initial delay for an agent can change the final agreement point to which the team converges and Figure 4.1c shows one example where the update period and delay of agent 1 has been modified. Figure 4.1d demonstrates that the choice of initial delay can change the minimal achievable distance to the adversary's desired goal $\mathbf{x}_g$. In addition, it shows that sometimes (but not always) larger delays are beneficial, so the system exhibits Neglect Benevolence. Figure 4.1e shows that changing the update periods of different agents can have dramatically different effects in our system.

Curiously, this example shows that the best choice of agent to influence is not the one whose initial state begins closest to the value for the desired goal (i.e. agent 3), nor the one whose initial state begins furthest from the desired goal (i.e. agent 7). Instead, changing the update period of agent 1 or agent 2 seems to be the most effective way to minimize the distance between the team's agreement point and the desired goal. In addition, introducing an initial delay reduces the required change in the update period of the influenced adversarial agent.

4.3 Computing Optimal Periods and Delays

In this section, we first develop some insight into the properties of the dynamical system described by the agent update rule in Equation (4.2) by analyzing its convergence properties. We then apply these results to reformulate the problem described in Section 4.2 to more efficiently compute optimal periods and delays via a simple algorithm.

4.3.1 Convergence Properties

Let us first consider case where each agent's state consists of a single real number (i.e. $m = 1$). The recurrence relation for the agent updates given in Equation (4.2) can be expressed as follows for some time-varying right stochastic matrix $\mathbf{D}\left(t, T_1, \ldots, T_n, \tau_1, \ldots, \tau_n\right)$ whose entries are a function of time t, the update periods $\{T_i \mid \forall i \in \mathcal{I}\}$ and the delays $\{\tau_i \mid \forall i \in \mathcal{I}\}$.

$$\mathbf{x}\left(t+1\right) = \mathbf{D}\left(t, T_1, \ldots, T_n, \tau_1, \ldots, \tau_n\right) \mathbf{x}\left(t\right) \tag{4.4}$$

Due to limited space, we will drop the dependence on update periods and delays from our notation and express this matrix as $\mathbf{D}(t)$. Entries of this time-varying matrix are given as follows, where $\mathbf{I}$ is the appropriately size identity matrix and $\mathbf{A}$ is the adjacency matrix capturing the neighbor information for the agents on the team.

$$D_{ij}(t) = \begin{cases} \frac{1}{\sum_{j=1}^{n} A_{ij}} A_{ij} & \text{if } i \in U(t), t \geq \tau_i \\ I_{ij} & \text{otherwise} \end{cases} \tag{4.5}$$

Note that each element of $\mathbf{D}(t)$ can only take on three different values (0, 1 or $\frac{1}{|\mathcal{N}_i|}$) over its time evolution. We can now write the recurrence relation in closed form as follows.

$$\mathbf{x}(t+1) = \mathbf{D}(t) \ldots \mathbf{D}(1) \mathbf{D}(0) \mathbf{x}(0) \tag{4.6}$$

To analyze the long-term (i.e. $t \to \infty$) behavior of this system, we first consider the simple case where all agents begin updating immediately without any delay (i.e. $\forall i : \tau_i = 0$). In this case, it is clear that the matrix $\mathbf{D}(t)$ actually takes on the same values periodically since each element is directly a function of the agent update periods. Taking $T_{\text{LCM}} = \text{lcm}(\{T_i \mid \forall i \in \mathcal{I}\})$, it is clear that the period of the matrix $\mathbf{D}(t)$ is T_{LCM}. By defining

$$\mathbf{P}(t) = \mathbf{D}(t) \ldots \mathbf{D}(1) \mathbf{D}(0) \tag{4.7}$$

we can write the time evolution of the joint state of the agents as follows.

$$\begin{aligned} \mathbf{x}((k+1) T_{\text{LCM}}) &= \mathbf{P}(T_{\text{LCM}} - 1) \mathbf{x}(k) \\ &= [\mathbf{P}(T_{\text{LCM}} - 1)]^{k+1} \mathbf{x}(0) \end{aligned} \tag{4.8}$$

For a given set of update periods, $\mathbf{P}(T_{\text{LCM}} - 1)$ is constant, so we have effectively taken our asynchronous system, where each agent executes its update rule with a different frequency, and written it as an equivalent synchronous system.

Theorem 2. *If the graph $\mathcal{G}$ representing the interaction topology for the multi-agent team contains a directed spanning tree, then the matrix $\mathbf{P}(T_{LCM} - 1)$ is an irreducible, aperiodic, right stochastic matrix.*

Proof. Since each matrix $\mathbf{D}(t)$ is right stochastic, the product of these matrices is also right stochastic. Each unique matrix $\mathbf{D}(t)$ is the transition matrix of a different Markov chain. Some of these Markov chains may be reducible or periodic, but each has a corresponding representation as a directed graph. However, it has been shown in [33] that if the union of the directed graphs corresponding to the transition matrices contains a directed spanning tree, then the product of those matrices will always result in a transition matrix that is irreducible and aperiodic. If the

graph $\mathcal{G}$ representing the interaction topology for the multi-agent team contains a directed spanning tree, then it is clear that the transition matrix $\mathbf{D}(T_{\text{LCM}} - 1)$ must also have a corresponding graph with a directed spanning tree. Thus, $\mathbf{P}(T_{\text{LCM}} - 1)$ must be the transition matrix for an irreducible and aperiodic Markov chain. □

Theorem 3. *If the graph $\mathcal{G}$ representing the interaction topology for the multi-agent team contains a directed spanning tree, then by applying the update rule in Equation* (4.2)*, the agents achieve global consensus to the agreement point $\mu^\top \mathbf{x}(0)$, where μ is the stationary distribution of $\mathbf{P}(T_{LCM} - 1)$.*

Proof. Since $\mathbf{P}(T_{\text{LCM}} - 1)$ is irreducible and aperiodic (Theorem 2), it must have an eigenvalue 1 with corresponding stochastic left eigenvector $\mu \in \mathbb{R}^n : \mu^\top \mathbf{1} = 1$. In addition, the largest eigenvalue of this matrix is unique and it is 1.

$$\begin{aligned} &\lim_{k\to\infty} \mathbf{x}(kT_{\text{LCM}}) \\ =&\lim_{k\to\infty} \left[\mathbf{P}(T_{\text{LCM}} - 1)\right]^k \mathbf{x}(0) \\ =&\left(\lim_{k\to\infty} \left[\mathbf{P}(T_{\text{LCM}} - 1)\right]^k\right) \mathbf{x}(0) \\ =&\mathbf{1}\mu^\top \mathbf{x}(0) \end{aligned} \tag{4.9}$$

Thus, the team of agents converges to the agreement point stated above. □

Remark 1. *Note that all of the convergence results above still apply when there are delays prior to the first update. Simply consider the properties of the following matrix.*

$$\tau_{max} = \max(\{\tau_i \mid \forall i \in \mathcal{I}\}) \tag{4.10}$$

$$\mathbf{P}(t) = \mathbf{D}(t)\,\mathbf{D}(t-1)\ldots\mathbf{D}(\tau_{max}) \tag{4.11}$$

$$\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{1}\mu^\top \mathbf{x}(\tau_{max}) \tag{4.12}$$

Then the agreement point of the system will be given by $\mu^\top \mathbf{x}(\tau_{max})$.

Remark 2. *Note that all of the convergence results above still apply when each agent's state consists of more than one element (i.e. $m \geq 2$). If $\mathbf{x}_i(t) \in \mathbb{R}^m$ is the state of agent i at time t, then let $\mathbf{x}_i^{(k)}(t) \in \mathbb{R}^m$ for $1 \leq k \leq m$ represent the k-th component of that agent's state. In addition, let the vector $\mathbf{x}^{(k)}(t) \in \mathbb{R}^n$ represent the k-th components of the states of all agents. Then the agreement point of the system will be given by $\left[\mathbf{x}^{(1)}(\tau_{max}) \quad \ldots \quad \mathbf{x}^{(m)}(\tau_{max})\right]^\top \mu$.*

4.3.2 Problem Reformulation and Algorithmic Solution

Based on the convergence results in the previous section, the problem statement in Equation (4.3) may be equivalently rewritten as the following non-linear integer program.

$$
\begin{aligned}
\underset{\forall i \in \mathcal{C}: T_i, \tau_i}{\arg\min} \quad & \left\| \mathbf{x}_g - \begin{bmatrix} \mathbf{x}^{(1)}\left(\tau_{\text{max}}\right) & \dots & \mathbf{x}^{(m)}\left(\tau_{\text{max}}\right) \end{bmatrix}^{\top} \mu \right\|_2^2 \\
\text{subject to} \quad & T_{\text{LCM}} = \text{lcm}\left(\{T_i \mid \forall i \in \mathcal{I}\}\right) \\
& \mathbf{P}\left(T_{\text{LCM}} - 1\right) = \mathbf{D}\left(T_{\text{LCM}} - 1 + \tau_{\text{max}}\right) \dots \mathbf{D}\left(\tau_{\text{max}}\right) \\
& \mathbf{P}\left(T_{\text{LCM}} - 1\right)^{\top} \mu = \mu \\
& \mu^{\top} \mathbf{1} = 1 \\
1 \leq k \leq m: \quad & \mathbf{x}^{(k)}\left(\tau_{\text{max}}\right) = \mathbf{D}\left(\tau_{\text{max}} - 1\right) \dots \mathbf{D}\left(1\right) \mathbf{D}\left(0\right) \mathbf{x}^{(k)}\left(0\right) \\
\forall i \in \mathcal{C}: \quad & 0 \leq \tau_i \leq \tau_{\text{upper}} \\
\forall i \in \mathcal{C}: \quad & 1 \leq T_i \leq T_{\text{upper}}
\end{aligned}
\tag{4.13}
$$

Noting that general integer programming is NP-hard, we now present an algorithm to find optimal solutions to the problem in Equation (4.13). Unlike the formulation in Equation (4.3), the formulation in Equation (4.13) does not require forward simulation of the system dynamics until convergence to find the agreement point to evaluate candidate solutions. This enables us to more efficiently evaluate candidate solutions making this difficult problem more tractable for applications.

Consider the known set $\mathcal{C} \subseteq \mathcal{I}$ of agents that can be influenced. Let the agents that can be influenced be given as $\mathcal{C} = \left\{c_1, c_2, \dots, c_{|\mathcal{C}|}\right\}$ where $\forall k : 1 \leq c_k \leq n$. Consider the tuple $S = \left(T_{c_1}, T_{c_2}, \dots, T_{|\mathcal{C}|}, \tau_{c_1}, \tau_{c_2}, \dots, \tau_{|\mathcal{C}|}\right)$. The first $|\mathcal{C}|$ elements of this tuple can take on values in the range $T_{c_k} \in \{1, 2, \dots, T_{\text{upper}}\}$ and the next $|\mathcal{C}|$ elements can take on values in the range $\tau_{c_k} \in \{0, 1, \dots, \tau_{\text{upper}}\}$. Each possible value of this tuple S represents a candidate solution for the problem given in Equation (4.13). We evaluate all possible values of this tuple S and select the one that results in the lowest value for our objective function. We can terminate early if we find S^* that results in the objective function being minimized below an acceptable threshold ε for a 'good enough' solution. Based on the convergence analysis presented in the previous section, the objective function is relatively inexpensive to evaluate and can be computed directly from the stationary distribution μ and $\forall k : \mathbf{x}^{(k)}\left(\tau_{\text{max}}\right)$. The worst-case time complexity of this algorithm is $O\left(\left(T_{\text{upper}} \tau_{\text{upper}}\right)^{|\mathcal{C}|}\right)$, which is exponential in the number of agents that can be influenced.

4.4 Applications

In this section, we present some illustrative simulations that demonstrate how asynchrony in consensus can be exploited by an adversary applying the techniques in the previous sections to achieve a variety of different goals. All simulations were conducted on a computer with an Intel Core i5 750 (4 cores, 2.66 GHz) processor and 12 GB RAM.

4.4.1 Multi-Robot Ambush

A common task for a multi-robot team in a hostile environment is for all members of the team to meet at a common location using only local interactions between neighboring robots and without advance knowledge of the meeting point. In swarm robotics, this behavior is more commonly known as rendezvous (or aggregation) and is often achieved by applying the discrete-time consensus update laws with the positions ($m = 2$ or $m = 3$) of the robots as the state variables. When the consensus updates are applied synchronously, it is well known that the rendezvous location (the agreement point) will be the centroid of their initial positions. When the consensus updates are performed asynchronously but the interaction topology remains connected (i.e. every update suggested by the topology occurs within finite time), the agents will converge to a common location, but the location may not be the centroid of their initial positions. A hostile adversary that can influence the periods and delays of the robots' consensus updates can potentially force the entire multi-robot team to converge in a dangerous location (e.g. where they would be ambushed).

Figure 4.2a shows the interaction topology for a multi-robot team of six robots. Figure 4.2b shows the trajectories and rendezvous location of the team when there is no adversarial influence. The robots' initial positions are shown by triangles and the dangerous region (unbeknownst to the robots) is shown as the yellow circle. Assume the adversary can influence only robot 4. As shown in Figure 4.2c, by selecting $\mathbf{x}_g$ as the center of the dangerous region and solving Equation (4.13), the adversary can compute the optimal delay and update period for robot 4 (i.e. $\tau_4 = 10$, $T_4 = 4$) to force the multi-robot team to converge as close as possible to the center of the dangerous region. For $\tau_{\text{upper}} = 12$ and $T_{\text{upper}} = 12$, the optimal solution was computed in 1.6 seconds on the computer. Figure 4.2d shows the trajectories resulting from the optimal choice of delays and periods when the adversary can influence both robot 2 and robot 4 (i.e. $\tau_2 = 0$, $T_2 = 2$, $\tau_4 = 8$, $T_4 = 9$). In this case, again with $\tau_{\text{upper}} = T_{\text{upper}} = 12$, the optimal solution was computed within 255 seconds on the computer, showing the exponential scaling in computation required as the number of influenced robots grows. However, only a marginally better solution

was found.

4.4.2 Maximally Diverting Opinions in Social Networks

There has been substantial work on convergence (or divergence) of opinions in social networks. For example, [1] discusses the idea of regular agents and forceful agents, such as the media, which can spread "misinformation". Given a network of social agents who update their opinions asynchronously, we show that the techniques described in this chapter could be used by an outside actor to influence a subset of agents to maximally divert the converged opinions from the expected agreement point by exploiting only asynchrony in updates. Given that $\mathbf{x}_{\text{orig}}$ is the expected agreement point with no outside influence, our objective can be rewritten as follows (all constraints still apply).

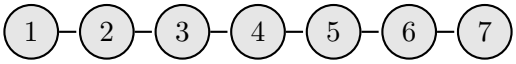
$$\underset{\forall i \in \mathcal{C}: T_i, \tau_i}{\arg\max} \left\| \mathbf{x}_{\text{orig}} - \begin{bmatrix} \mathbf{x}^{(1)}\left(\tau_{\max}\right) & \dots \mathbf{x}^{(m)}\left(\tau_{\max}\right) \end{bmatrix}^{\top} \mu \right\|_2^2 \tag{4.14}$$

Figure 4.3a shows the interaction topology for this group of agents. Figure 4.3b shows the natural evolution of the opinions of these agents on a particular binary topic (agent opinions are confined to the range 0 to 1). Figure 4.3b shows the evolution of agent opinions when an adversary optimally influences agent 5 by changing its delay and update period to cause maximum diversion of agent opinions from the original agreement point. As expected, the larger the size of the group, the more challenging it is to divert opinions with only a single adversarially influenced agent.

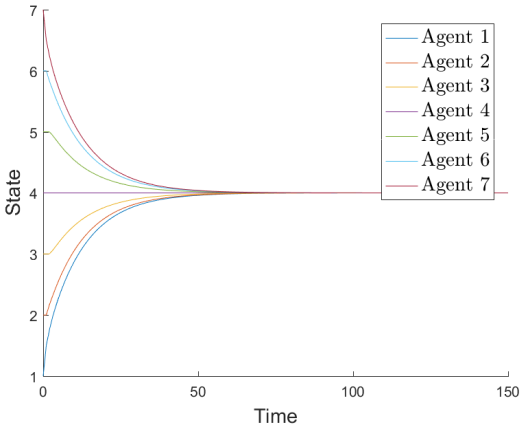
4.5 Conclusion

In this chapter, we considered a team of agents that updated their states using the discrete-time consensus protocol, but with different update frequencies and delays prior to first update. Given a subset of agents on the team that we could influence, we studied the problem of changing only their update periods and delays to change to the agreement point reached by the team. An analysis of the convergence properties of the system led to a reformulation that enabled more efficiently evaluating candidate solutions (i.e. without naive forward simulation of system dynamics) to optimally select the update periods and delays to minimize the distance between the final agreement point and a desired goal. In illustrative simulations, we explored how this technique could be used by an adversary to change the agreement point of a multi-robot team performing rendezvous and force the robots to rendezvous in a dangerous region of the map. In

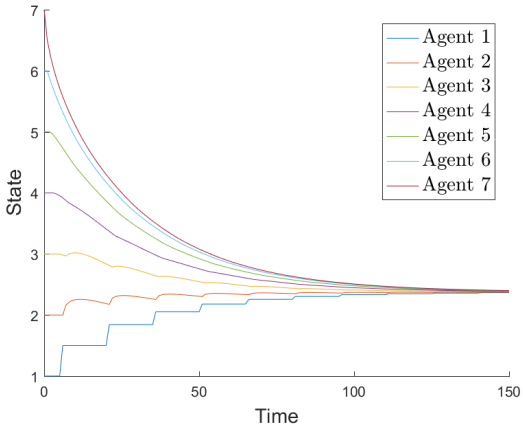
another application, we demonstrated that it was possible to select the update periods and delays to cause a group of social agents to converge to an agreement point as far as possible from the original agreement point. In future work, we plan to use heuristics to quickly find approximate solutions to our problem, further improving tractability for practical applications. We also plan to investigate mitigation techniques.



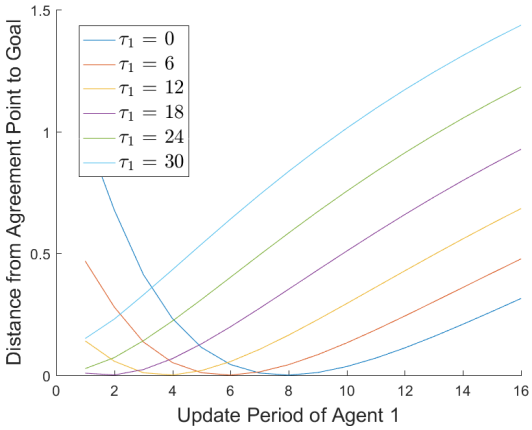
(a) Interaction topology for an example team of seven agents.



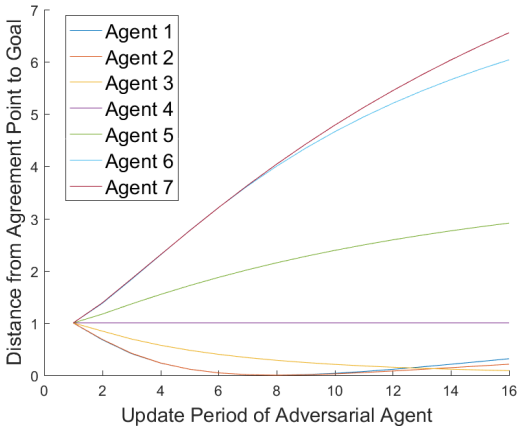
(b) State evolution of multi-agent team with default update periods. The agents converge to an agreement point 4.



(c) State evolution of multi-agent team when $\tau_1 = 5$ and $T_1 = 15$. The agents converge to an agreement point 2.3751.

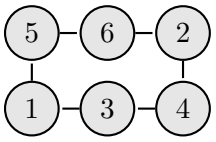


(d) Assume only agent 1 can be influenced. The choice of delay τ_1 changes the agreement point and the minimum achievable distance to the desired goal.

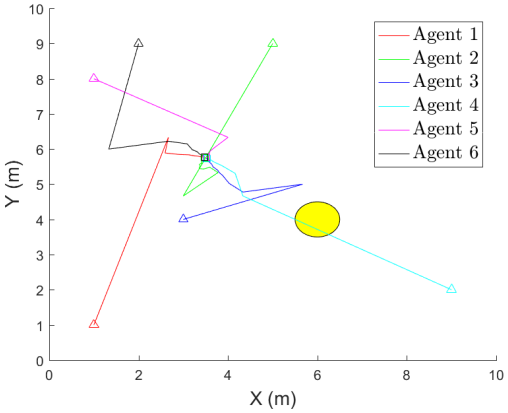


(e) Assume only one of the agents can be influenced. The choice of adversarial agent changes the minimum achievable distance to the desired goal.

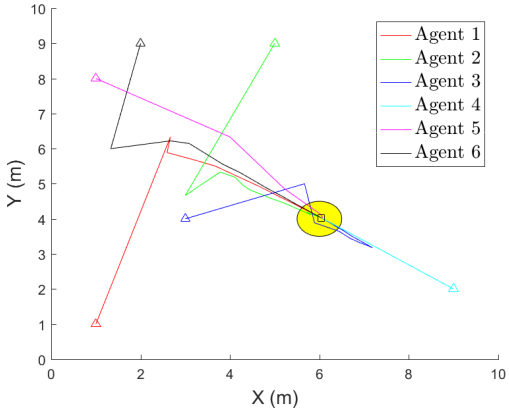
Figure 4.1: A team of seven agents has the interaction topology given by the graph shown in (a). The joint initial state of the team is $\mathbf{x}(0) = [1, 2, 3, 4, 5, 6, 7]^\top$. The default update periods are $\forall i : T_i = 1$ and initial delays are $\forall i : \tau_i = 0$. The adversary's desired goal is $\mathbf{x}_g = 3$.



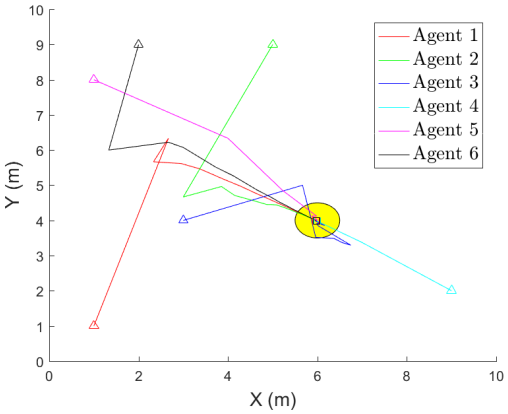
(a) Interaction topology for a robot team.



(b) State evolution of multi-robot team with default update periods. The agents converge to an agreement point $[3.501, 5.755]^\top$.

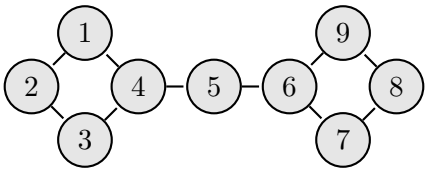


(c) State evolution of multi-robot team when $\tau_4 = 10$ and $T_4 = 4$. The agents converge to an agreement point $[6.0348, 4.0331]^\top$.

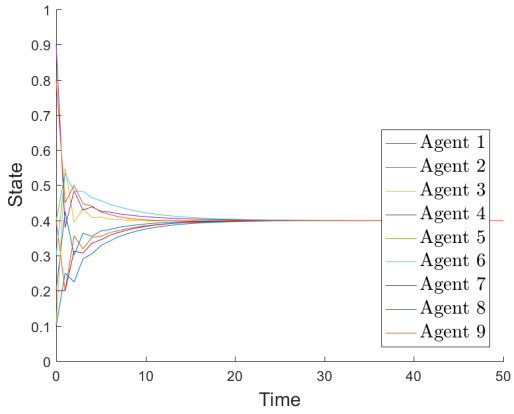


(d) State evolution of multi-robot team when $\tau_2 = 0$, $\tau_4 = 8$, $T_2 = 2$ and $T_4 = 9$. The agents converge to an agreement point $[5.9797, 3.9890]^\top$.

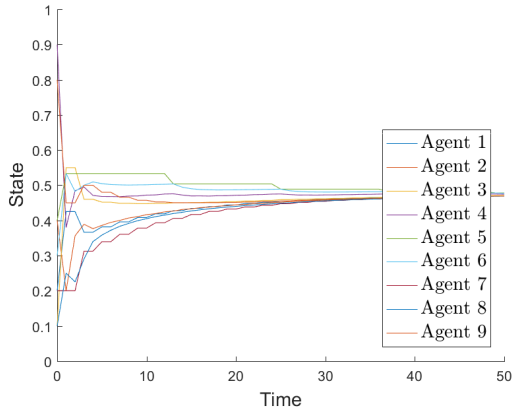
Figure 4.2: A robot team has the interaction topology shown in (a). The initial robot positions are shown as triangles. The default delays are $\forall i : \tau_i = 0$ and update periods are $T_1 = 2, T_2 = 1, T_3 = 1, T_4 = 3, T_5 = 4, T_6 = 2$. The adversary's desired goal is $\mathbf{x}_g = [6, 4]^\top$.



(a) Interaction topology for some social agents.



(b) State evolution of agents with default update periods. The agents converge to an agreement point 0.4275.



(c) State evolution of agents when $\tau_5 = 0$ and $T_5 = 12$. The agents converge to an agreement point 0.4748.

Figure 4.3: A group of social agents has the interaction topology shown in (a). The initial agent opinions are $\mathbf{x}(0) = [0.2, 0.4, 0.1, 0.9, 0.4, 0.3, 0.2, 0.1, 0.8]^\top$. The default delays are $\forall i : \tau_i = 0$ and update periods are $T_1 = 2, T_2 = 1, T_3 = 2, T_4 = 1, T_5 = 2, T_6 = 1, T_7 = 2, T_8 = 1, T_9 = 2$.

Chapter 5

Automated Behavior Sequencing with Known Switch Times

Swarm robotics is characterized by using simple robots in large numbers to accomplish complex tasks. The emergent behaviors based on the local interactions among the robots and their environment allows the swarm to accomplish tasks even with the sensor and computation limitations necessary to produce a large group of robots inexpensively. Unfortunately, the algorithms producing the emergent behaviors are often only guaranteed to succeed in very controlled environments (e.g. obstacle-free environments), which is not practical in real-world applications. Real-world applications include area coverage, search and rescue operations, military surveillance or even first responder assistance [7, 34, 42, 45, 62]. The environments in which robotic swarms are used for these applications are often cluttered and non-convex. One way to achieve complex tasks with swarms is to synthesize local control laws that would accomplish the task while satisfying practical constraints like collision avoidance [80]. However, this is a hard problem in general and there are no known solutions with guaranteed performance for tasks like area coverage, search and rescue, and surveillance [44]. Alternatively, one can use a library of simple swarm behaviors (which may not necessarily be designed for the task at hand) and compose them using a supervisory controller to accomplish the task. In this chapter, we study the problem of swarm behavior composition to achieve complex tasks.

More formally, the problem of swarm behavior composition can be stated as follows: *Given a library of swarm behaviors, $\mathcal{B}$, and an objective function or performance criterion encoding the task at hand, find the sequence of behaviors and the times at which the behaviors should be switched to accomplish the desired task.* Although behavior composition for accomplishing complex tasks has been widely studied for single robot systems [4, 9], it is not well studied

for swarm systems. The key features of the swarm robotic systems that make the behavior composition problem challenging are (a) the dimension of the joint state space of the swarm is very high, (b) very often the closed loop dynamics of the individual robots using the local control laws are nonlinear and so it is hard to have an a priori estimate of the state of the system at a future time, (c) convergence guarantees for behaviors are available only under restrictive assumptions like no collision avoidance and hence the swarm may not converge to the desired behavior. The above features makes it difficult to evaluate the effect of executing a behavior for a prescribed time and hence planning a sequence of behaviors becomes difficult.

In this chapter, we solve a simpler version of the general behavior composition problem where we assume that the behaviors are selected at given discrete decision time points. After formalizing the problem, we present an informed search algorithm that computes the optimal behavior sequence that should be executed from a given initial state to maximize swarm performance on the task. We then apply our algorithm to a swarm navigation application in which our library of behaviors is well-suited for the task objectives. Conversely, we then apply our algorithm to a swarm dynamic area coverage application, which is an application for which none of the behaviors in our library has been originally designed. This demonstrates the utility of our algorithm in composing behaviors to accomplish tasks for which individual behaviors were not originally designed.

Finally, we note that in prior work, experiments on human control of robotic swarms have identified challenges and interesting phenomena such as Neglect Benevolence [79], where delaying an input can actually increase overall system performance. These discoveries led to the formalization of the phenomenon, which resulted in an algorithm for identifying the optimal input time [57]. Interestingly, in returning to human-swarm interaction experiments, this algorithm enabled the creation of a rigorous benchmark task for evaluating human performance in estimating the optimal input time [58], something that humans need to do in real-world situations where automated algorithms may not be sufficiently efficient. Existing experiments in human-swarm interaction [44] have repeatedly identified the challenge human operators face in selecting appropriate swarm behaviors. We believe the algorithm we present, which is proven to be both *optimal* and *complete* for the given decision time points, not only represents a contribution (a) to solving the problem of choosing an optimal behavior sequence, but also (b) as an enabling factor in the creation of additional benchmarks for human-swarm interaction performance.

5.1 Related Work

While many different types of collective behaviors for swarms have been studied in the literature [12, 44], it is somewhat difficult to find a consistent mathematical definition of swarm behavior within prior work. Simple local control laws have been proposed to generate spatially coordinated motion for groups of robots for swarm behaviors like flocking [69], rendezvous [3, 23] and pattern formation [6]. Many of these swarm behaviors are often parameterized and small changes in parameters can drastically change the overall system dynamics and the swarm's ability to accomplish a task. The tuning of these parameters for a particular task — even in the rare event that the task can be completed with only one behavior — is another significant challenge. To address these issues, we first provide a consistent mathematical definition of a swarm behavior that allows us to treat very different behaviors like flocking and pattern formation in the same framework. In addition, we differentiate between *swarm meta-behaviors* defined by a set of control laws with open parameters (e.g. flocking attraction radius), and *concrete swarm behaviors*, which are instantiated from swarm meta-behaviors by specifying the parameters resulting in fixed swarm dynamics. Thus, it is possible to systematically generate a behavior library by instantiating concrete swarm behaviors from swarm meta-behaviors described by existing control laws in the literature.

5.2 Formalization of the Swarm Behavior Sequencing Problem

We begin by considering a remote robotic swarm that is controlled by a supervisory operator (e.g. a human). There are a variety of tasks (e.g. cohesive navigation through a cluttered environment, area coverage) that the operator would like to complete with the robots in the swarm, but the supervisory operator cannot interact directly with individual robots and instead can only control the swarm in aggregate. The operation of individual robots is defined by local interactions with other robots in their spatial neighborhood and the global choice of swarm behavior. More accurately, the choice of swarm behavior defines how each robot reacts to local information (sensed or communicated) from neighboring robots. There is a fixed library of swarm behaviors and the operator interacts with the entire swarm by selecting a particular behavior from the library. Note that each behavior in the library is actually the result of every robot in the swarm executing a particular local control law corresponding to that behavior. The resulting local interactions give rise to overall system dynamics, which we define as a swarm behavior. A different local control law will give rise to a different swarm behavior. In this way, each robot can be programmed with

a fixed number of local controllers and the operator's global choice of behavior selects which local control law is executed by all of the robots. Once a behavior has been selected, it is active over a certain time interval after which a new behavior may be selected. The operator may choose to execute the same behavior again if desired. In such a system, the operator selects a sequence of swarm behaviors to complete a task and it is desirable to select the "best" sequence of swarm behaviors. However, in many real-world scenarios, it is often unclear whether there even exists a behavior sequence that enables the swarm to complete the task. Thus, our goal is to automate this process for the operator. More specifically, given a performance criterion for a particular task the operator would like to complete, our objective is to automatically choose the optimal sequence of behaviors to complete the task and optimize the performance criterion. In addition, if it is not possible to complete the task using the behaviors in the library, we would like to identify this situation and report it.

5.2.1 Mathematical Formulation

We treat the robotic swarm as a distributed dynamical system where the swarm consists of M robots and the state of an individual robot i at time t is given by $\mathbf{x}^{(i)}(t) \in \mathbb{R}^N$. The joint state of the swarm at time t is given by $\mathbf{x}(t) \in \mathbb{R}^d$ where $d = MN$. There is a finite-sized, fixed library

$$\mathcal{B} = \{f_1, f_2, \ldots, f_B\} \tag{5.1}$$

of swarm behaviors, where each behavior is a piecewise continuous map

$$f_j : \mathbb{R}^d \to \mathbb{R}^d \tag{5.2}$$

representing the overall system dynamics that govern the joint state evolution of the swarm. There is a time horizon $t_f \in \mathbb{R}_+$ over which we would like to use the swarm to complete a particular task (e.g. move to an area). In order to complete the task, we would like to select a sequence of behaviors from the library. There is an ordered set

$$\mathcal{T} = \{t_0, t_1, t_2, \ldots, t_n\} \tag{5.3}$$

of n decision time points $t_k \in \mathbb{R}_+$ at which we can select a new behavior to be executed by the robotic swarm. Without loss of generality assume $t_0 = 0$. For $k \in \{0, 1, \ldots, n\}$, the state evolution of the swarm over each time interval

$$T_k = [t_k, t_{k+1}) \tag{5.4}$$

is governed by the choice of behavior

$$b_k \in \{1, 2, \ldots, |\mathcal{B}|\} \tag{5.5}$$

that is selected for that interval. Note that the final time interval is from the last decision time point to the end of the time horizon (i.e. $T_n = [t_n, t_f)$). Specifically, during time interval T_k, the state of swarm evolves according to the following differential equation

$$t \in [t_k, t_{k+1}) : \dot{\mathbf{x}}(t) = f_{b_k}(\mathbf{x}(t)) \tag{5.6}$$

where $\dot{\mathbf{x}}(t)$ is the derivative of the joint state with respect to time. The state of the swarm at the end of time interval T_k is given by

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \int_{t_k}^{t_{k+1}} f_{b_k}(\mathbf{x}(\tau))\, d\tau \tag{5.7}$$

with $\mathbf{x}(t_0) = \mathbf{x}_0$. Given a performance criterion, our objective is to choose a behavior sequence $S = (b_0, b_1, b_2, \ldots, b_n)$ that optimizes the performance criterion. Our performance criterion can be expressed as a cost function $P(\mathbf{x}_0, \mathcal{X}_g, \mathcal{B}, \mathcal{T}, S)$ we want to minimize with the following properties, where $\mathcal{X}_g$ is a set of goal states for our swarm and $S_i \sqcup S_j$ represents the concatenation of sequences S_i and S_j. The first property assigns an infinite cost to all behavior sequences that do not cause the swarm to reach some $\mathbf{x} \in \mathcal{X}_g$ from x_0. The second property states that all behavior sequences S executed from $\mathbf{x}_0$ beginning with a particular prefix subsequence S_i can be computed in two parts: the cost of the sequence S_i executed from $\mathbf{x}_0$ plus the cost of the remaining sequence S_j executed from $\mathbf{x}_{S_i}$ (i.e. the state reached by executing S_i from $\mathbf{x}_0$ at times $\mathcal{T}_{S_i}$). Thus, if we can accurately estimate the cost of S_j using a heuristic, we can estimate the cost of the entire sequence S.

$$\forall S : \mathbf{x}(t_f) \notin \mathcal{X}_g \implies P(\mathbf{x}_0, \mathcal{X}_g, \mathcal{B}, \mathcal{T}, S) = \infty \tag{5.8}$$

$$P(\mathbf{x}_0, \mathcal{X}_g, \mathcal{B}, \mathcal{T}, S_i \sqcup S_j) = P(\mathbf{x}_0, \mathcal{X}_g, \mathcal{B}, \mathcal{T}_{S_i}, S_i) + P\left(\mathbf{x}_{S_i}, \mathcal{X}_g, \mathcal{B}, \mathcal{T}_{S_j}, S_j\right) \tag{5.9}$$

5.3 Algorithm for Behavior Sequencing

Given the initial state $\mathbf{x}_0$, a set of goal states $\mathcal{X}_g$, a library of swarm behaviors $\mathcal{B}$, time horizon t_f, set of decision time points $\mathcal{T}$, a swarm behavior execution cost function $C(\cdot)$ and heuristic cost-to-goal estimation function $H(\cdot)$, Algorithm 1 finds a sequence of swarm behaviors selected from $\mathcal{B}$ to apply at times in $\mathcal{T}$ to ensure that the swarm is in one of the goal states in $\mathcal{X}_g$ at the end of the time horizon t_f. If such a swarm behavior sequence does not exist, then the algorithm simply returns an empty sequence ($\varnothing$) to indicate failure. Note that the set of goal states $\mathcal{X}_g$ does not need to be specified explicitly, but given a particular state $\mathbf{x}$ it must be possible to check whether $\mathbf{x} \in \mathcal{X}_g$ or $\mathbf{x} \notin \mathcal{X}_g$. Algorithm 1 may be described as an informed search algorithm similar in spirit to traditional search algorithms such as best-first search and the many variants

of A*. The algorithm searches over the space of swarm behavior sequences by incrementally building the sequence and choosing to evaluate sequences with a lower estimated cost first.

The algorithm begins on line 2 by initializing the empty priority queue Q, setting the best known swarm behavior sequence S_{best} to $\varnothing$ and setting the cost c_{best} of the best known sequence to ∞. The priority queue Q will contain a set of 3-tuples with each tuple consisting of (a) a state $\mathbf{x}_s$, (b) the swarm behavior sequence S_s used to reach state $\mathbf{x}_s$ from the initial state $\mathbf{x}_0$ and (c) the cost c_s to reach state $\mathbf{x}_s$ by applying the partial swarm behavior sequence S_s. In addition, all of the tuples in the queue are ordered by a priority p_s, which is the estimated minimum cost of the complete behavior sequence to reach $\mathcal{X}_g$ from $\mathbf{x}_0$ if the complete sequence must begin with the partial sequence S_s. On line 3, this estimate of the minimum cost of the best swarm behavior sequence from $\mathbf{x}_0$ to reach a state in $\mathcal{X}_g$ is computed using the heuristic function $H(\cdot)$ and stored as p_0. On line 4, the search is initiated by enqueuing on Q with priority p_0 the 3-tuple containing the initial state $\mathbf{x}_0$, an empty partial swarm behavior sequence (i.e. $\varnothing$) and initial cost (i.e. zero). While the queue is not empty, lines 5–26 are executed to expand each partial swarm behavior sequence in the queue.

On line 6, the tuple with lowest priority (i.e. estimated minimum cost to reach $\mathcal{X}_g$) is dequeued from the queue. Assuming that our heuristic function never overestimates the minimum cost required to reach $\mathcal{X}_g$ (i.e. it is admissible), then if the dequeued tuple has a higher estimated cost p_s than the cost c_{best} of the best sequence found so far (line 7), it is clearly not possible for the search to produce a better sequence, so we can terminate the search early and return S_{best} as the best swarm behavior sequence (line 8). Otherwise, we proceed to evaluate more swarm behavior sequences.

One line 10, we identify the length k of the current partial swarm behavior sequence S_s. If the length of the sequence is equal to the number of decision time points (line 11), then the sequence is complete (i.e. no further expansion is possible because we have reached the end of the time horizon) and we can evaluate the sequence. The swarm behavior sequence is feasible if the goal has been reached (i.e. $\mathbf{x}_s \in \mathcal{X}_g$). If swarm behavior sequence is feasible and has a lower cost than any known sequence (line 12), then we store it as the best known sequence S_{best} and update the minimum cost c_{best} (line 13).

Conversely, if the length of the partial swarm behavior sequence S_s is less than the number of decision time points (line 15), then it can be expanded by selecting another behavior for the next time interval $[t_k, t_{k+1})$. On lines 16–24, the partial sequence S_s is expanded using each behavior in the library $\mathcal{B}$. On line 17, the successor state $\mathbf{x}_e$ reached by applying behavior f_j over time interval $[t_k, t_{k+1})$ is computed. While the definite integral may be evaluated numerically, in some cases it is possible to compute the solution analytically (e.g. linear time-invariant system

Algorithm 1 Find the Best Behavior Sequence

function FINDBESTSEQUENCE($\mathbf{x}_0, \mathcal{X}_g, \mathcal{B}, t_f, \mathcal{T}, C, H$)
  $Q \leftarrow \varnothing$, $S_{\text{best}} \leftarrow \varnothing$, $c_{\text{best}} \leftarrow \infty$
  $p_0 \leftarrow H\left(\mathbf{x}_0, \mathcal{X}_g, \mathcal{B}, t_f, \mathcal{T}, 0\right)$
  ENQUEUE($Q, p_0, (\mathbf{x}_0, \varnothing, 0)$)
  while $Q \neq \varnothing$ **do**
    $p_s, (\mathbf{x}_s, S_s, c_s) \leftarrow$ DEQUEUE(Q)
    if $c_{\text{best}} \leq p_s$ **then**
      return S_{best}
    end if
    $k \leftarrow |S_s|$
    if $k = |\mathcal{T}|$ **then**
      if $(\mathbf{x}_s \in \mathcal{X}_g) \wedge (c_s < c_{\text{best}})$ **then**
        $S_{\text{best}} \leftarrow S_s$, $c_{\text{best}} \leftarrow c_s$
      end if
    else if $k < |\mathcal{T}|$ **then**
      for all $f_j \in \mathcal{B}$ **do**
        $\mathbf{x}_e \leftarrow \mathbf{x}_s + \int_{t_k}^{t_{k+1}} f_j\left(\mathbf{x}\left(\tau\right)\right) d\tau$
        $S_e \leftarrow S_s \sqcup (j)$
        $c_e \leftarrow c_s + C\left(\mathbf{x}_s, f_j, t_{k+1} - t_k\right)$
        $p_e \leftarrow c_e + H\left(\mathbf{x}_e, \mathcal{X}_g, \mathcal{B}, t_f, \mathcal{T}, k+1\right)$
        if $p_e < c_{\text{best}}$ **then**
          ENQUEUE($Q, p_e, (\mathbf{x}_e, S_e, c_e)$)
        end if
      end for
    end if
  end while
  return S_{best}
end function

dynamics), which can greatly speed up computation. On line 18, the partial sequence S_s is concatenated with the swarm behavior index j to generate the successor sequence S_e. On line 19, the cost of executing behavior f_j for a time period of length $t_{k+1} - t_k$ is computed using the provided cost function $C(\cdot)$ and added to the cost c_s of the current sequence to give the cost of the successor sequence c_e. Note that the cost function $C(\cdot)$ also implicitly captures environmental details and enables us to identify and handle situations where selecting a behavior results in a potentially invalid trajectory through state space for the robotic swarm. For example, if the state trajectory from the current state $\mathbf{x}_s$ to successor state $\mathbf{x}_e$ as result of behavior f_j causes some robots in the swarm to collide with an obstacle, then the cost function $C(\cdot)$ would evaluate to ∞ indicating that S_e is an invalid partial swarm behavior sequence. That invalid sequence would then no longer be expanded. For valid sequences, on line 20, the estimated cost of reaching $\mathcal{X}_g$ from $\mathbf{x}_e$ over time period $[t_{k+1}, t_f)$ is computed and added to c_e to generate a priority p_e for sequence S_e. Finally, on line 22, the tuple containing the successor state $\mathbf{x}_e$, sequence S_e and cost c_e is enqueued with priority p_e.

Since the algorithm is very similar in spirit to A* search, the algorithm inherits many of its properties. These properties include the fact that the algorithm is complete and optimal with respect to the inputs (e.g. decision time points) when (a) the cost function is monotonically non-decreasing with respect to the length of the sequence (i.e. adding a behavior to the sequence can never decrease the cost of the sequence) and (b) the heuristic function is admissible (i.e. it never overestimates the minimum cost to reach $\mathcal{X}_g$).

Theorem 4. *If the cost function is monotonically non-decreasing, the heuristic is admissible and a feasible swarm behavior sequence exists, then Algorithm 1 will find this sequence. Otherwise, it will return an empty sequence $\varnothing$ to indicate failure. That is, Algorithm 1 is complete.*

Proof. First, note that if lines 7–9 are omitted, then there is no method to exit the loop on lines 5–26 until the queue is empty and thus the algorithm will necessarily evaluate all possible swarm behavior sequences of length $|\mathcal{T}|$. Since it evaluates all possible behavior sequences, if a feasible sequence exists, the algorithm must find it. In addition, note that as the algorithm proceeds, since the heuristic is admissible, the lowest priority in the queue is also actually a lower bound on the minimum cost to reach $\mathcal{X}_g$ via any possible partial sequence that can be expanded from the queue. On line 7, c_{best} will be ∞ unless a feasible solution has been found and all feasible solutions must have $c_{\text{best}} < \infty$. Thus, if a feasible solution has not been found, line 8 will only be reached if $c_{\text{best}} = p_s = \infty$. In that case, the algorithm will still terminate correctly on line 8 and $S_{\text{best}} = \varnothing$ will be returned to indicate that no feasible solution exists. □

Theorem 5. *If the cost function is monotonically non-decreasing and the heuristic is admissible,*

then Algorithm 1 will find the optimal behavior sequence to minimize the provided cost function.

Proof. As noted in the proof for Theorem 4, c_{best} is ∞ unless a feasible sequence has been found. From Theorem 4, if a feasible sequence exists, it will be found. In addition, since the lowest priority in the queue is a lower bound on the minimum cost for any sequence expanded from the queue, it is clear that line 8 will only be reached if the lowest priority in the queue becomes larger than c_{best}. In this case, no better solution can possibly exist, so the algorithm has certainly found the optimal swarm behavior sequence with the minimum cost. □

Since Algorithm 1 can evaluate all possible swarm behavior sequences in the worst case, it has a worst-case time complexity $O\left(|\mathcal{B}|^{|\mathcal{T}|}\right)$. As usual for most informed search algorithms, the actual real-world performance of the algorithm depends on how accurately the heuristic $H(\cdot)$ estimates the minimum cost of the best swarm behavior sequence.

Based on the above results, and similarly to Weighted A*, we note that Algorithm 1 can also incorporate a multiplicative weight on the admissible heuristic resulting in bounded suboptimality (with respect to total cost) of the solution swarm behavior sequence. Line 20 would be modified as follows, where $w \in \mathbb{R}_+ : w \geq 1$ is a chosen weight and the swarm behavior sequence found by the algorithm will have a cost at most w times the cost of the optimal sequence.

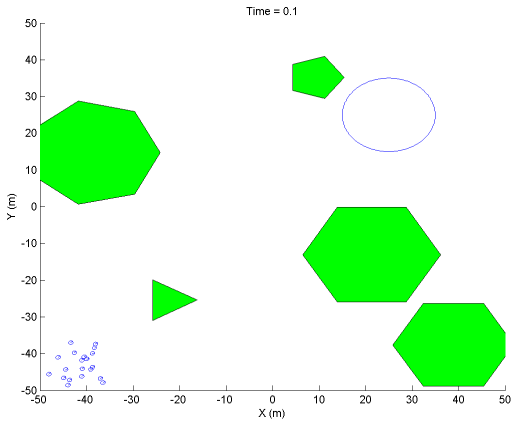
$$p_e \leftarrow c_e + wH\left(\mathbf{x}_e, \mathcal{X}_g, \mathcal{B}, t_f, \mathcal{T}, k+1\right) \tag{5.10}$$

In [2], a clever Multi-Heuristic variant of A* is given, in which it is shown that multiple arbitrarily inadmissible heuristics can be combined with a single admissible heuristic to make the search more computationally efficient while maintaining the properties of bounded suboptimality and completeness. For applications where multiple heuristics are available, Algorithm 1 could also be made more computationally efficient by incorporating the techniques from [2].

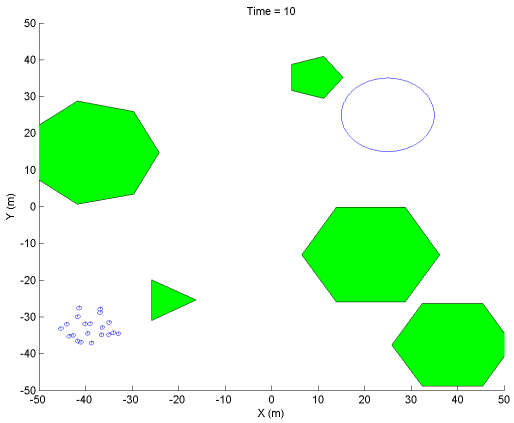
5.4 Simulated Robotic Swarm

5.4.1 Robot Dynamic Model

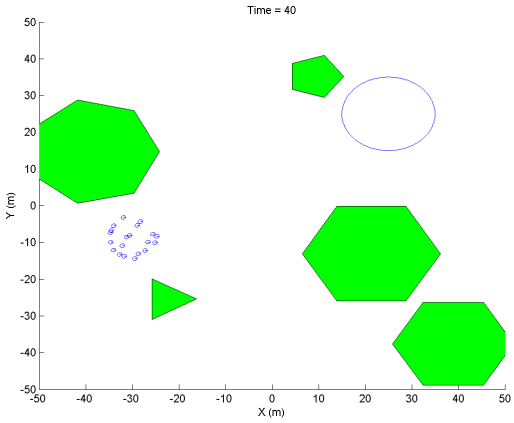
Each robot in the swarm has the following dynamic model, where x^i, y^i and θ^i are the state variables representing the position and orientation of robot i. The control inputs to the robot are given by u_v^i and u_ω^i, which represent the commanded linear velocity and commanded angular



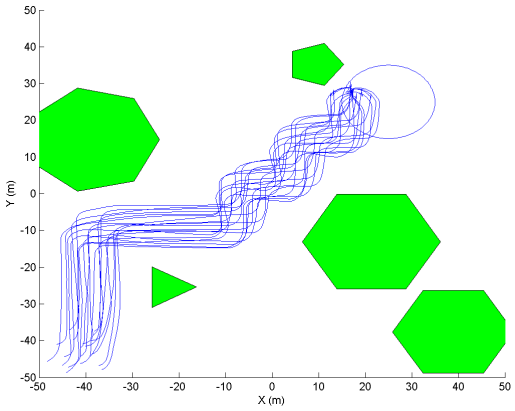
(a) Initial Robot Poses



(b) Poses After of 1^{st} Behavior (Flock North) in Sequence over 1^{st} Time Interval



(c) Poses After of 4^{th} Behavior (Flock East) in Sequence over 4^{th} Time Interval



(d) Trajectory of the Swarm

Figure 5.1: Simulated Swarm of 20 Robots Executing a Behavior Sequence to Move to a Target Area

velocity.

$$\begin{aligned} \dot{x}^i &= u_v^i \cos\left(\theta^i\right) \\ \dot{y}^i &= u_v^i \sin\left(\theta^i\right) \\ \dot{\theta}^i &= u_\omega^i \end{aligned} \tag{5.11}$$

For convenience, we define the position vector $\mathbf{p}^i \in \mathbb{R}^2$, which is the xy-coordinates of the robot as given above, and bearing vector $\mathbf{b}^i \in \mathbb{R}^2 : \|\mathbf{b}^i\|_2 = 1$, which is a unit vector in the heading direction θ^i. The function $\phi(\mathbf{v}_1, \mathbf{v}_2)$ finds the smallest angle required to rotate from $\mathbf{v}_1$ to $\mathbf{v}_2$. Assume $\mathbf{v}_1$ to $\mathbf{v}_2$ are augmented appropriately (with zeros) to compute their cross product and that $\hat{\mathbf{e}}_3$ is the unit vector normal to the x-y plane along the positive z-axis.

$$\begin{aligned} \mathbf{b}^{ij} &= \frac{\mathbf{p}^j - \mathbf{p}^i}{\|(\mathbf{p}^j - \mathbf{p}^i)\|_2} \\ \phi(\mathbf{v}_1, \mathbf{v}_2) &= \operatorname{sgn}\left((\mathbf{v}_1 \times \mathbf{v}_2)^T \hat{\mathbf{e}}_3\right) \cos^{-1}\left(\frac{\mathbf{v}_1^T \mathbf{v}_2}{\|\mathbf{v}_1\|_2 \|\mathbf{v}_2\|_2}\right) \end{aligned} \tag{5.12}$$

Within the simulation, our inputs are bounded ($u_v^i \in [U_{v,\min}, U_{v,\max}]$, $u_\omega^i \in [U_{\omega,\min}, U_{\omega,\max}]$) and may saturate as would be expected for a real robotic swarm. The chosen saturation limits for our simulated swarm were $U_{v,\max} = -U_{v,\min} = 2$ and $U_{\omega,\max} = -U_{\omega,\min} = \frac{\pi}{8}$.

5.4.2 Swarm Meta-Behaviors

In this section, we introduce a set of meta-behaviors for our robotic swarm. These meta-behaviors are parameterized and generate different dynamics for our robotic swarm depending on the choice of parameters. Specifying the parameters for these meta-behaviors allows us to instantiate the meta-behaviors into concrete behaviors (see Section 5.4.3) that would be included in the behavior library $\mathcal{B}$.

Biased Flocking

The following control law generates flocking behavior for the swarm but with flocking biased in the direction specified by the parameter $\mathbf{q} \in \mathbb{R}^2$. Within the equations below, $\mathcal{N}_r(i)$ specifies neighbors of robot i within the repulsion radius R_r, $\mathcal{N}_h(i)$ specifies neighbors outside of R_r but within the heading alignment radius R_h, and $\mathcal{N}_a(i)$ specifies neighbors outside of R_h but within the attraction radius R_a. Finally, $\mathcal{N}(i)$ specifies robots within any of the three regions.

The minimum forward velocity for all robots was $U_{v,\text{default}}$.

$$\begin{aligned}
\mathbf{v}^{(i)} =& \frac{1}{|\mathcal{N}(i)|}\left(\sum_{j\in\mathcal{N}_r(i)}\frac{-\mathbf{b}^{ij}}{\|\mathbf{p}^{(j)}-\mathbf{p}^{(i)}\|_2^2}+\sum_{j\in\mathcal{N}_a(i)}\left(\mathbf{p}^{(j)}-\mathbf{p}^{(i)}\right)\right)\\
\gamma^{(i)} =& \frac{1}{|\mathcal{N}(i)|}\left(\sum_{j\in\mathcal{N}_r(i)}\phi\left(\mathbf{b}^i,-\mathbf{b}^{ij}\right)+\sum_{j\in\mathcal{N}_h(i)}\phi\left(\mathbf{b}^i,\mathbf{b}^j\right)+\sum_{j\in\mathcal{N}_a(i)}\phi\left(\mathbf{b}^i,\mathbf{b}^{ij}\right)\right)\\
u_v^i =& \max\left(\left(\left(\mathbf{v}^{(i)}+\mathbf{q}\right)^T\mathbf{b}^i\right),U_{v,\text{default}}\right)\\
u_w^i =& \gamma^{(i)}+\phi\left(\mathbf{b}^i,\mathbf{q}\right)
\end{aligned} \tag{5.13}$$

The chosen radii for our simulated swarm were $R_r = 5$, $R_h = 10$, $R_a = 20$ and the chosen minimum forward velocity was $U_{v,\text{default}} = 1$.

Rendezvous / Anti-Rendezvous

The following control law can generate different dynamics depending on the choice of parameter K_p. If $K_p > 0$, then each robot moves toward neighboring robots. If $K_p < 0$, each robot moves away from neighboring robots. If $K_p = 0$, then each robot stops moving.

$$\begin{aligned}
\dot{\mathbf{p}}^{(i)} &= \frac{K_p}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\left(\mathbf{p}^{(j)}-\mathbf{p}^{(i)}\right)\\
u_v^i &= \left(\mathbf{b}^i\right)^T\dot{\mathbf{p}}^{(i)}\\
u_w^i &= \phi\left(\mathbf{b}^i,\dot{\mathbf{p}}^{(i)}\right)
\end{aligned} \tag{5.14}$$

Formation Control

For formation control, we use the following consensus-based control law from [57]. The formation itself is specified via the vector $\mathbf{z}\in\mathbb{R}^{2M}$ (note that this was called K in [57]). The formation may be specified using a set of positions in the vector $\mathbf{z}$ and the following consensus-based control law will generate the desired formation around the centroid of robots' initial positions. Thus, the formation can be specified independently of the robots' actual positions in the workspace. If the robot inputs don't saturate, the control law below preserves the centroid of the robots' initial positions as they move and also ensures that *differences* between robot positions in their final configuration will match the differences between their positions in the vector $\mathbf{z}$. Note that $\mathbf{z}$ is

the only parameter to this meta-behavior.

$$\begin{aligned}\dot{\mathbf{p}}^{(i)}(t) =& \frac{1}{|\mathcal{N}(i)|}\left(\sum_{j\in\mathcal{N}(i)}\left(\mathbf{p}^{(j)}-\mathbf{p}^{(i)}\right)-\sum_{j\in\mathcal{N}(i)}\left(\mathbf{z}^{(j)}-\mathbf{z}^{(i)}\right)\right)\\ u_v^i =& \left(\mathbf{b}^i\right)^T\dot{\mathbf{p}}^{(i)}\\ u_w^i =& \phi\left(\mathbf{b}^i,\dot{\mathbf{p}}^{(i)}\right)\end{aligned} \tag{5.15}$$

5.4.3 Swarm Behaviors

In this section, we describe a set of concrete swarm behaviors. These behaviors have no unspecified parameters, so the dynamics that result from these control laws are fully specified. These are behaviors that would be included in the behavior library $\mathcal{B}$ for Algorithm 1.

Flock

All robots flock in a direction that is a result of their initial positions and orientations. Use control law in Section 5.4.2 with bias $\mathbf{q} = \mathbf{0}$.

Flock East

Robots flock in the positive x-direction. Use control law in Section 5.4.2 with bias $\mathbf{q} = \begin{bmatrix}1 & 0\end{bmatrix}^T$.

Flock North

Robots flock in the positive y-direction. Use control law in Section 5.4.2 with bias $\mathbf{q} = \begin{bmatrix}0 & 1\end{bmatrix}^T$.

Flock West

Robots flock in the negative x-direction. Use control law in Section 5.4.2 with bias $\mathbf{q} = \begin{bmatrix}-1 & 0\end{bmatrix}^T$.

Flock South

Robots flock in the negative y-direction. Use control law in Section 5.4.2 with bias $\mathbf{q} = \begin{bmatrix}0 & -1\end{bmatrix}^T$.

Stop Moving

All robots stop moving. Use control law in Section 5.4.2 with $K_p = 0$.

Rendezvous

All robots move toward each other. Use control law in Section 5.4.2 with $K_p = 1$.

Anti-Rendezvous

All robots move away from each other. Use control law in Section 5.4.2 with $K_p = -1$.

Line X

Robots move into a line parallel to the x-axis. The control law in Section 5.4.2 is used with $\mathbf{z}^{(0)} = \mathbf{0}$ and $\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} + \begin{bmatrix} 2 & 0 \end{bmatrix}^T$.

Line Y

Robots move into a line parallel to the y-axis. The control law in Section 5.4.2 is used with $\mathbf{z}^{(0)} = \mathbf{0}$ and $\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} + \begin{bmatrix} 0 & 2 \end{bmatrix}^T$.

5.5 Application to Swarm Navigation

In many real-world scenarios, a supervisory operator for a robotic swarm would like to complete a mission composed of many tasks. A very common task in many missions is for the supervisory operator to navigate the robotic swarm from an initial area to a target area through a region that may contain obstacles. For example, consider a robotic swarm that is transported on a larger carrier vehicle. It may not be possible for the carrier vehicle to reach the region where the swarm must operate for the remainder of the mission. The carrier vehicle would drop off the swarm in the initial area and it would be necessary for the supervisory operator to navigate the swarm to the target area to complete the rest of the mission. While there has been a tremendous amount of work on both complex multi-robot motion planning and distributed multi-robot control, here we explore a slightly different approach where we use Algorithm 1 to select a sequence of behaviors

that moves the robots from the initial region to the target region without colliding with obstacles. Note that this approach has the benefit of only selecting from behaviors corresponding to local control laws already programmed onto the robots and it only requires the operator to transmit a short sequence of behavior identifiers and time points (i.e. a few integers and floating point numbers) to the robot swarm, resulting in very little data transmission between the operator and the swarm. Therefore, this approach is very useful for applications in domains where the bandwidth between the human supervisor and the swarm is very limited, for example where there is a swarm of robots under water supervised by a human on a ship.

For this scenario, we use the simulated robotic swarm described in Section 5.4. The state $\mathbf{x}$ is composed of positions $\mathbf{p}^i$ and orientations θ^i of all robots in the swarm. Given a target region specified by center $\mathbf{a}_{\text{target}} \in \mathbb{R}^2$ and radius $R_{\text{target}} \in \mathbb{R}_+$, the set of goal states $\mathcal{X}_g$ is given as follows.

$$\mathcal{X}_g = \left\{ \mathbf{x} \mid \forall i : \left\| \mathbf{p}^i - \mathbf{a}_{\text{target}} \right\|_2 < R_{\text{target}} \right\} \tag{5.16}$$

Our cost function $C(\cdot)$ penalizes the motion of each robot in the swarm. This ensures that our optimal behavior sequence navigates the swarm to the target area while minimizing the motion of all robots. It is defined as

$$C(\mathbf{x}_s, f, t) = \sum_i L_f \left(\mathbf{p}^i(0), \mathbf{p}^i(t) \right) \tag{5.17}$$

where L_f computes the length of the curve from $\mathbf{p}^i(0)$ to $\mathbf{p}^i(t)$ when each individual robot trajectory $\mathbf{p}^i(t)$ is based on the joint state trajectory $\mathbf{x}(t) = \mathbf{x}_s + \int_0^t f(x(\tau))\, d\tau$. Note that we define $L_f(\mathbf{p}^i(0), \mathbf{p}^i(t)) = \infty$ if the trajectory from $\mathbf{p}^i(0)$ to $\mathbf{p}^i(t)$ intersects an obstacle or moves off the map. Our heuristic estimates the cost of the optimal behavior sequence using the traditional Euclidean distance metric and sums the straight-line distances between each robot and the target area. This is the minimum distance each robot would need to travel if it were possible for the robot to drive directly toward the target area, so the heuristic will never overestimate the cost of a behavior sequence. Thus, the heuristic is admissible.

$$H(\mathbf{x}_s, \mathcal{X}_g, \mathcal{B}, t_f, \mathcal{T}, k) = \sum_i \max\left(\left(\left\| \mathbf{p}^i - \mathbf{a}_{\text{target}} \right\|_2 - R_{\text{target}} \right), 0 \right) \tag{5.18}$$

Figure 5.1 shows a simulation where Algorithm 1 was applied to move the swarm from the initial area in the bottom left corner of the map to the target area shown by the blue circle while avoiding the green obstacles over a time horizon of $t_f = 150$ seconds. There were 15 evenly spaced decision time points $0 \leq k \leq 14 : t_k = 10k$ seconds. The map was a square area with $X \in [-50, 50)$ meters and $Y \in [-50, 50)$ meters For this simulation, the heuristic had a multiplicative weight of $w = 2$ leading to bounded suboptimality (i.e. the solution sequence had

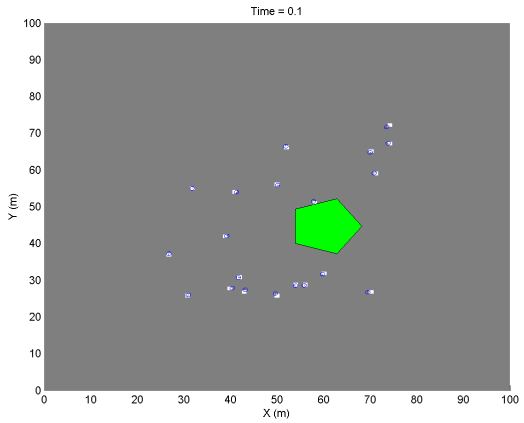
no greater than 2 times the cost of the best sequence). The chosen swarm behavior sequence was Flock North, Flock North, Flock North, Flock East, Flock East, Flock East, Flock North, Flock East, Flock North, Flock East, Flock North, Stop Moving, Stop Moving, Stop Moving, Rendezvous. Since no cost was placed on time-to-completion, there were several intervals in the middle where the Stop Moving behavior was chosen rather than having the swarm Stop Moving after reaching the target area. In addition, since no cost was placed on switching swarm behaviors across adjacent time intervals, the chosen sequence alternates between Flock North and Flock East more frequently than necessary. Adding a cost on time-to-target and switching swarm behaviors can make the behavior sequence match human intuition more closely. Most other parts of the sequence match intuition: Flock North and Flock East were selected to navigate to the target area that was northeast of the robot start positions and Rendezvous was selected at the end because the centroid (but not all robots) of the swarm was within the target area. In that state, Rendezvous clearly had a lower motion cost than Flock East.

Experiments were conducted on a desktop with an Intel Core i5 750 (2.66 GHz, quad-core) processor and 12GB of RAM. Across 10 trials with 20 robots in environments with different arrangements of obstacles, the average time to identify the behavior sequence with $w = 2$ was 27.92 seconds. When the environment was kept constant and $w = 2$, for 10, 15, 20 and 25 robots, it took 9.19, 15.98, 24.52 and 35.42 seconds respectively to find the best sequence. For the $w = 10$, the corresponding times were slightly faster at 9.52, 15.62, 19.27 and 24.63 seconds.

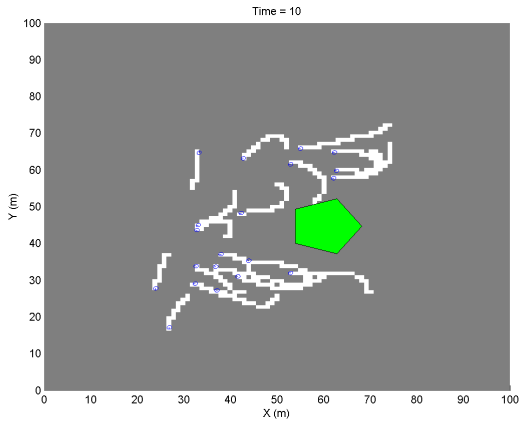
5.6 Application to Swarm Dynamic Area Coverage

In this section, we explore the use of Algorithm 1 to find the best sequence of swarm behaviors to complete a task that the *behaviors were not originally designed to accomplish*. In contrast to the swarm navigation task described in Section 5.5 where the goal of motion towards a target area mapped well to swarm behaviors such as biased flocking that achieve motion in a coherent direction, here we try to complete a dynamic area coverage task over a given time horizon using the same library of behaviors — none of which have been designed in advance to maximize the area covered by the swarm. Specifically, given an environment that has been discretized into a number of grid cells, we try to find a swarm behavior sequence that ensures a certain percentage of grid cells have been visited by a robot in the swarm within the given time horizon. This is in contrast to static area coverage where robots with a certain sensing region (e.g. disk, cone) try to move to fixed final poses that maximize the area simultaneously viewed by all robots.

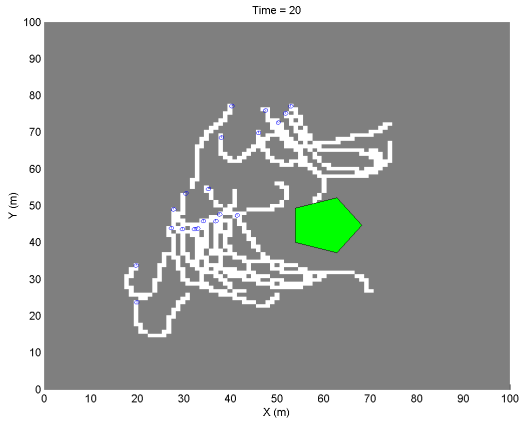
We use the simulated robotic swarm described in Section 5.4, but for this scenario, we augment



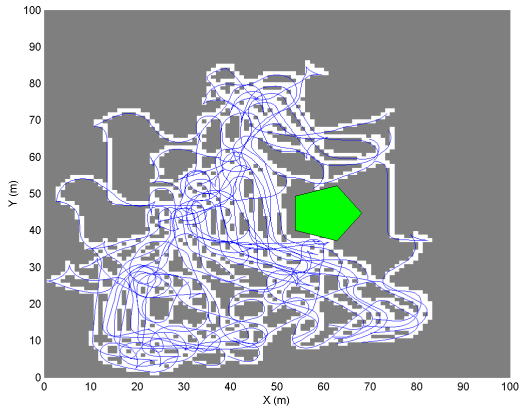
(a) Initial Robot Poses



(b) Poses After 1st Behavior (Flock West) in Sequence over 1st Time Interval



(c) Poses After 2nd Behavior (Flock North) in Sequence over 2nd Time Interval



(d) Trajectory of the Swarm Over Complete Time Horizon

Figure 5.2: Simulated Swarm of 20 Robots Executing a Behavior Sequence to Achieve at Least 25% Area Coverage While Minimizing Robot Motion

our state $\mathbf{x}$ with the state of the grid cells. Each grid cell g_{xy} has a corresponding binary variable in the state $\mathbf{x}$ where g_{xy} is 1 if it has been visited by a swarm robot and 0 if it has not. The set of goal states $\mathcal{X}_g$ is given as follows. Assume the grid has dimensions $G_x \times G_y$. Here R_{target} is the desired coverage ratio.

$$\mathcal{X}_g = \left\{ \mathbf{x} \mid \sum_x \sum_y g_{xy} \geq R_{\text{target}} G_x G_y \right\} \tag{5.19}$$

Our cost function $C(\cdot)$ again penalizes the motion of each robot in the swarm.

$$C(\mathbf{x}_s, f, t) = \sum_i L_f \left(\mathbf{p}^i(0), \mathbf{p}^i(t) \right) \tag{5.20}$$

This time our heuristic estimates the cost of the optimal behavior sequence by the minimum distance the robots would need to travel to cover the remaining cells that need to be visited to achieve the desired coverage ratio if the cells were arranged in the best possible configuration. Note that M is the number of robots in the swarm.

$$H(\mathbf{x}_s, \mathcal{X}_g, \mathcal{B}, t_f, \mathcal{T}, k) = \frac{G_s\sqrt{2}}{3} \max \left(0, R_{\text{target}} G_x G_y - 4M - \sum_x \sum_y g_{xy} \right) \tag{5.21}$$

Assume each grid cell is square with side length G_s. The cost computed by the heuristic is the minimum distance each robot would have to travel to cover enough cells to achieve the desired coverage ratio if all the remaining cells to be covered were divided (not necessarily evenly) among the robots in the swarm and it was possible for each swarm robot to cover its respective cells by driving diagonally in an almost straight line by zig-zagging very slightly. In this way, it would be possible for each robot to visit $3\alpha + 4$ cells for $\alpha \in \mathbb{N}$ by travelling only $(\alpha + \epsilon) G_s\sqrt{2}$ units, where $\epsilon \in \mathbb{R}_+$ can be made arbitrarily small. Thus, this heuristic is admissible and will never overestimate the minimum cost to visit the remaining cells.

Figure 5.2 shows a simulation where Algorithm 1 was applied to find a swarm behavior sequence that achieves at least 25% area coverage (i.e. $R_{\text{target}} = 0.25$) while minimizing the distance travelled by the robots over a time horizon of $t_f = 150$ seconds. The map was a square area with $X \in [0, 100)$ meters and $Y \in [0, 100)$ meters and square grid cells with side length $G_s = 1$ meter. There were 15 evenly spaced decision time points $0 \leq k \leq 14 : t_k = 10k$ seconds. For this simulation, the heuristic had a multiplicative weight of $w = 5$ leading to bounded suboptimality (i.e. the solution sequence had no greater than 5 times the cost of the best sequence). The chosen swarm behavior sequence was Flock West, Flock North, Flock South, Anti-rendezvous, Line X, Rendezvous, Stop Moving, Stop Moving, Stop Moving, Flock South, Flock South, Flock East, Line X, Flocking, Flock West. We note that a wider variety of swarm behaviors were chosen to

accomplish this task than the navigation task. Initially, the robots were very dispersed and could not Rendezvous without colliding with the obstacle. While one might expect a lawn mower pattern to emerge given the flocking behaviors in the library, this is not the sequence that was chosen. In retrospect, this is logical since all of the flocking behaviors are somewhat wasteful of individual robot motion with multiple robots revisiting the same grid cells as they move in the same direction. Thus, a more diverse combination of behaviors in the sequence led to a lower overall motion cost to achieve the desired coverage ratio.

Experiments were conducted on the same computer used for Section 5.5. Across 10 trials with 20 robots in environments with different arrangements of obstacles, the average time to identify the behavior sequence when $w = 5$ was 63.55 seconds. When the environment was held fixed and the weights were $w = 10$, $w = 30$, $w = 50$ and $w = 70$ the time required was 50.49, 29.43, 19.98 and 18.75 seconds respectively verifying that higher bounds on suboptimality enabled solutions to be found more quickly though the gains eventually diminished. A comparison between sequences varying the number of robots did not make sense for this application because over the same time horizon, the area covered was highly correlated with the number of robots.

5.7 Conclusion

We formalized the swarm behavior sequencing problem, the solution of which enables a supervisory operator to select the optimal behavior sequence for a swarm to complete a given task. We presented an algorithm that performs an informed search over sequences of behaviors to find the optimal swarm behavior sequence. The algorithm was shown to be optimal and complete for the chosen decision time points. We applied the algorithm to (a) swarm navigation, where the swarm behaviors in the library mapped well to the required task and (b) swarm area coverage, which none of the swarm behaviors in the library were individually designed to accomplish. Simulation results in static environments with obstacles showed that using an admissible weighted heuristic, our algorithm successfully selected the best behavior sequences with bounded suboptimality to accomplish the task. While the execution times for the algorithm were too high for real-time supervisory control of a real robot swarm, the optimality (or bounded suboptimality) and completeness guarantees of the algorithm validate its usefulness for creating benchmarks for human performance. It can also be used to periodically aid a human operator when required. Execution times can be significantly improved in the future through the use of parallel computing hardware such as GPUs.

Chapter 6

Time-Optimal Scheduling of Consensus-based Behaviors to Achieve Multiple Goals

Swarms are multi-robot systems that operate using simple local control laws. Global swarm behaviors emerge via interactions of swarm members. These emergent behaviors, such as flocking, deployment, rendezvous, allow the swarm to accomplish tasks even with the individual swarm members' sensor and computation limitations. Real-world applications, including area coverage, search and rescue operations, military surveillance or first responder assistance [7, 34, 42, 45, 62], are composed of complex tasks which usually cannot be achieved with a single existing behavior. One way to achieve complex tasks with swarms is to manually synthesize more sophisticated local control laws that would allow the swarm to accomplish the task while individual robots satisfy practical local constraints like collision avoidance [11, 80]. Other work has used formal methods to automatically synthesize the local control logic for each robot from a set of individual robot specifications [51], which provides formal guarantees for each robot, but limited insight into overall swarm performance. This is a hard problem in general and there are few known solutions with guaranteed performance even for specific tasks like area coverage, search and rescue, and surveillance [12, 35, 44]. Alternatively, one can use a library of collective swarm behaviors (which may not necessarily be designed for the task at hand) and compose them using a supervisory controller [59] to accomplish the task. Often, an operator knows, not only the final task goal but also a set of intermediate goals that must be reached on the way to the final goal. In this chapter, we study the problem of finding a *behavior schedule* (i.e. behavior sequence and the times of instantiation of the behaviors), so that the total time to reach the final goal is minimized and all intermediate goals are achieved.

More formally, we consider a swarm robotic system equipped with a library of collective behaviors. In each behavior, the state evolution of the swarm is modeled by a linear dynamical system. Formally, we model the state evolution of the swarm as a hybrid dynamical system, where there is one mode with linear dynamics corresponding to each behavior. We are also given a set of intermediate goals (represented in the joint state space of the robots) that are unordered and a final goal that the robots should reach. Note that each intermediate goal as well as the final goal is the equilibrium point of some behavior in our behavior library, which ensures that the intermediate and final goals are reachable. Our objective is to compute a sequence of the modes (behaviors) and the switch times such that the robot swarm reaches the final goal state in minimum time while achieving all the intermediate goal states. Our problem is related to the problem of switch time optimization studied for hybrid systems [49]. In switch time optimization problems for hybrid systems, the objective is to compute the sequence of the modes and the switch times between the modes so as to optimize a given objective function (over the path taken by the dynamical system) over a given time horizon. A key distinction of our problem from the extant literature on switch time optimization is that we have intermediate states that the dynamical system has to reach. Furthermore, in our problem, the final state is fixed while the objective is to minimize the total time to get to the final goal state.

We develop a two-step procedure to solve the swarm behavior scheduling problem. Our first step is to compute the minimum time trajectory between any two given (possibly intermediate) goal states with our dynamics as constraints. We present an algorithm to compute the different behavior switches and their timings for moving from one goal state to another. Please note that it has already been established in the literature that for going to a given goal state in a time-optimal fashion, it may be beneficial to delay the choice of the behavior that results in achievement of the goal [57]. The result here is a generalization of the result in [57], since we show that one can use other behaviors (whose fixed points may not correspond to the given goals) transiently in order to move between the goals in a time-optimal fashion. Since our method is based on gradient descent, we achieve a locally optimal solution. In our second step, we formulate a Traveling Salesman Problem, where the cities are the goals and the cost of traveling between the goals is as given by the first step above. We show that the costs satisfy the triangle inequality although they are asymmetric. Therefore, we use a variant of Christofides' algorithm [21, 39] to compute a goal sequence with bounded suboptimality - specifically, it is a constant factor approximation to the optimal sequence.

Our contributions are (a) formalization of the problem of behavior scheduling to achieve multiple unordered goals with a robotic swarm, (b) an algorithm that produces a behavior schedule to achieve all intermediate goals and the final goal in minimum time such that the behavior durations are locally optimal and given which the goal sequence has bounded suboptimality and (c)

application of this algorithm to configuration control of robot swarms.

6.1 Related Work

We formulate our problem in the framework of switch time optimization for hybrid systems (see [49] for a comprehensive survey of recent results) where the mode sequence is not known a priori and also apply some results from traditional time-optimal control to characterize the length of the unknown optimal mode sequence. The authors of [32] consider the problem of finding the optimal switch times for a hybrid system where the cost functional is defined over a fixed time horizon and the final state is unconstrained. After finding an expression for the gradient of the cost functional in terms of the dynamics of the system and the costate, they propose a gradient descent algorithm that iteratively forward integrates the state equations, backward integrates the costate equations and then takes a gradient step (with Armijo step size) in the cost functional to eventually find a locally optimal solution. They also propose a procedure to use the gradient related information to iteratively alternate between adding modes to the sequence and updating the switch times. In [81], a sufficient descent version of the algorithm is presented with a procedure to satisfy dwell time constraints. The techniques presented in [81] are quite powerful and provide locally optimal solutions to the free-endpoint fixed-time horizon switch time optimization problem in hybrid systems with piecewise continuous modes.

In contrast to this locally optimal solution to the general free-endpoint fixed-time problem, in the first part of our approach we solve a particular case (for systems performing consensus) of the constrained-endpoint free-time problem using a more direct approach that does not require integration of the costate equations. Our method also relies on gradient descent techniques and provides a locally optimal solution. This solution is a behavior schedule for a robotic swarm to move from an initial state towards a goal state in minimum time using only consensus-based behaviors from a given library. In the second part of our approach, we then extend this technique with a variant [39] of Christofides' algorithm [21] for generating fixed-endpoint Hamiltonian paths with bounded suboptimality, which allows us to find a behavior schedule for a robotic swarm to move from an initial state to a desired final state while also achieving a set of unordered intermediate goals in minimum time. Our primary contribution is the application of these techniques to robotic swarms.

6.2 Problem Formulation

6.2.1 Consensus-based Behaviors

Consider a robotic swarm whose joint state (i.e. stacked states of individual robots) is given by $\mathbf{x}(t) \in \mathcal{X}$ at time $t \in \mathbb{R}_+$ in state space $\mathcal{X} = \mathbb{R}^n$. For this robotic swarm, the state evolution of each robot may be written as a weighted linear combination of its own state and the states of other robots in the swarm. That is, the joint state evolution of the swarm is given by the following differential equation when there is no external influence from a supervisory operator (e.g. a human).

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) \tag{6.1}$$

For the dynamics above, it is clear that the joint state of the swarm continuously evolves over time unless $\mathbf{x}(t)$ lies in the null space of the dynamics matrix $\mathbf{A}$. If the system is stable (i.e. all eigenvalues of $\mathbf{A}$ have negative real parts), then it is clear that the state of the system will evolve from any initial state $\mathbf{x}(0)$ to asymptotically approach a point in the null space of $\mathbf{A}$ (i.e. $\lim_{t\to\infty} \mathbf{A}\mathbf{x}(t) = \mathbf{0}$).

The state evolution of the swarm may be influenced by selecting a constant bias input $\mathbf{z} \in \mathbb{R}^n$, which is incorporated into the dynamics as follows.

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\mathbf{x}(t) - \mathbf{z}) \tag{6.2}$$

Applying the terminology we introduced in [59], Equation (6.2) represents the dynamics for a *swarm meta-behavior* with two unspecified parameters: the dynamics matrix $\mathbf{A}$ and the bias vector $\mathbf{z}$. This meta-behavior can be instantiated into many different *concrete swarm behaviors* via different choices for the dynamics matrix $\mathbf{A}$ and the bias vector $\mathbf{z}$. For example, if individual robot states represent their spatial positions, then the concrete swarm behavior known as rendezvous may be instantiated by specifying $\mathbf{z} = \mathbf{0}$ and $\mathbf{A} = -\mathbf{L}$, where $\mathbf{L}$ is the Laplacian matrix of the swarm's communication graph. In fact, the case when the dynamics matrix is fixed as $\mathbf{A} = -\mathbf{L}$ is an important subclass of the meta-behavior with the dynamics given in Equation (6.2). Specifically, fixing the dynamics matrix in this way results in the swarm performing a biased version of linear time-invariant (LTI) consensus with the following dynamics. See [68] for more information about the unbiased version of continuous-time continuous-state consensus.

$$\dot{\mathbf{x}}(t) = -\mathbf{L}(\mathbf{x}(t) - \mathbf{z}) \tag{6.3}$$

For this reason, we refer to this meta-behavior and any concrete behaviors instantiated from it (by specifying $\mathbf{z}$) as *consensus-based behaviors*. When the states represent spatial positions of

the robots, we can instantiate concrete consensus-based behaviors like rendezvous (as described above) or various spatial configurations (e.g. line, circle) through appropriate selection of $\mathbf{z}$. In contrast to other methods of rendezvous or generating the spatial configurations, instantiating them as a consensus-based behavior would provide useful properties such as preserving the centroid of the original robot positions as they move.

It is important to recognize that the states do not necessarily have to represent spatial positions. For example, the states could instead represent more abstract quantities, such as the distribution of robots among tasks as in [67] (continuous-time macroscopic model of swarm). In that case, we can instantiate concrete consensus-based behaviors such as mission-specific robot distributions among tasks by appropriately selecting $\mathbf{z}$ and with $\mathbf{A} = -\mathbf{L} = \mathbf{K}$ ($\mathbf{K}$ as defined in [67]) representing the transition rate matrix, where $\mathbf{L}$ (not necessarily symmetric) is the weighted Laplacian of a directed graph modelling transitions between tasks. Here, the consensus-based behavior would provide the beneficial (and necessary) property that the total quantity of robots is preserved as their distribution among tasks changes.

6.2.2 Problem Statement

Assume we have a finite-sized, fixed library

$$\mathcal{B} = \{f_1, f_2, \ldots, f_m\} \tag{6.4}$$

of m concrete consensus-based behaviors where the i-th behavior in the library is a map $f_i : \mathbb{R}^n \to \mathbb{R}^n$ that has been instantiated from Equation (6.3) by specifying the bias vector $\mathbf{z}_i \in \mathbb{R}^n$.

$$f_i\left(\mathbf{x}\left(t\right)\right) = -\mathbf{L}\left(\mathbf{x}\left(t\right) - \mathbf{z}_i\right) \tag{6.5}$$

For notation purposes, we write $\mathcal{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m\}$ for the set of bias vectors with which the concrete consensus-based behaviors in our library were instantiated.

A supervisory operator controls the robotic swarm by selecting a behavior $f_{b_k} \in \mathcal{B}$ where $b_k \in \{1, 2, \ldots, |\mathcal{B}|\}$ and applying it for a corresponding duration $\tau_k \in \mathbb{R}_+$. Thus, control of the robotic swarm may be represented as a sequence

$$S = \left(\left(b_1, \tau_1\right), \left(b_2, \tau_2\right), \ldots, \left(b_d, \tau_d\right)\right) \tag{6.6}$$

of d 2-tuples where the first element in the 2-tuple is the index b_k of the selected behavior in the library and the second element is the duration τ_k over which the behavior is applied. We refer to

this sequence as a *behavior schedule* for our robotic swarm. For notation purposes, we write the behavior selection times

$$t_k = t_0 + \sum_{i=1}^{k} \tau_i \tag{6.7}$$

and without loss of generality assume $t_0 = 0$. The corresponding dynamics guiding the state evolution of the swarm is given as follows.

$$\dot{\mathbf{x}}(t) = \begin{cases} f_{b_1}(\mathbf{x}(t)) & \text{for } t \in [0, t_1) \\ f_{b_2}(\mathbf{x}(t)) & \text{for } t \in [t_1, t_2) \\ \vdots & \\ f_{b_d}(\mathbf{x}(t)) & \text{for } t \in [t_{d-1}, t_d) \end{cases} \tag{6.8}$$

Our supervisory operator has an unordered set of c intermediate goals $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_c\}$ where $\mathbf{y}_j \in \mathbb{R}^n$ that they would like to achieve with the robotic swarm in minimal time. We define a goal $\mathbf{y}_j$ to be achieved within time T if the following condition is true.

$$\exists t \leq T : \|\mathbf{P}(\mathbf{x}(t) - \mathbf{y}_j)\|_2^2 \leq \varepsilon \tag{6.9}$$

The parameter $\varepsilon \in \mathbb{R}_+$ is a desired tolerance chosen by the supervisory operator. The matrix $\mathbf{P}$ is an orthogonal projection matrix that rejects any component in the null space of $\mathbf{L}$. That is, if $\left\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{\text{nullity}(\mathbf{L})}\right\}$ is an orthonormal basis (i.e. $\forall i : \|\mathbf{v}_i\|_2 = 1$ and $\forall i, j : \mathbf{v}_i^\top \mathbf{v}_j = 0$) for the null space of $\mathbf{L}$, then $\mathbf{P}$ can be written as follows.

$$\mathbf{P} = \prod_{i=1}^{\text{nullity}(\mathbf{L})} \left(\mathbf{I} - \mathbf{v}_i \mathbf{v}_i^\top\right) \tag{6.10}$$

For example, every Laplacian matrix $\mathbf{L}$ has an eigenvector $\mathbf{1}$ with corresponding eigenvalue 0. For a connected undirected graph, $\mathbf{1}$ is also the only eigenvector with eigenvalue 0, so it must be part of a basis for the null space of $\mathbf{L}$, which means for an undirected connected graph $\mathbf{P} = \mathbf{I} - \left(\mathbf{1}^\top \mathbf{1}\right)^{-1} \mathbf{1}\mathbf{1}^\top$.

In this chapter, given the library $\mathcal{B}$ of consensus-based behaviors, the initial state $\mathbf{x}_{\text{initial}}$ of the swarm, the final goal $\mathbf{x}_{\text{final}}$ and the unordered set $\mathcal{Y}$ of intermediate goals the operator would like to achieve, our objective is to identify the behavior schedule S, which consists of selected behaviors and the associated time intervals of application, to minimize the total time T. Formally,

our problem is written as follows.

$$
\begin{aligned}
\underset{S}{\arg\min} \quad & T \\
\text{subject to} \quad & \mathbf{x}(0) = \mathbf{x}_{\text{initial}} \\
& \left\|\mathbf{P}\left(\mathbf{x}(T) - \mathbf{x}_{\text{final}}\right)\right\|_2^2 \leq \varepsilon \\
& \dot{\mathbf{x}}(t) = \begin{cases} f_{b_1}(\mathbf{x}(t)) & \text{for } t \in [0, t_1) \\ f_{b_2}(\mathbf{x}(t)) & \text{for } t \in [t_1, t_2) \\ \quad \vdots & \\ f_{b_d}(\mathbf{x}(t)) & \text{for } t \in [t_{d-1}, t_d) \end{cases} \\
& \forall \mathbf{y} \in \mathcal{Y}, \exists t \leq T : \left\|\mathbf{P}\left(\mathbf{x}(t) - \mathbf{y}\right)\right\|_2^2 \leq \varepsilon
\end{aligned}
\tag{6.11}
$$

The first constraint in this problem specifies the initial state of the swarm. The second constraint specifies a necessary and sufficient condition on the final state of the swarm for the final goal to be achieved. The third constraint restricts the trajectory describing the state evolution of the swarm based on the dynamics of the chosen behaviors from the library. The final constraint not only captures the requirement that all intermediate goals must be achieved, but also that they may be achieved in any order. The optimal solution to this problem is a behavior schedule S^* that causes the swarm to achieve all intermediate goals and then the final goal in minimal time T^*.

To provide some mathematical intuition, this problem is visualized for a system with a two-dimensional ($n = 2$) state space and stable dynamics matrix (eigenvalues of $\mathbf{A}$ have negative real parts) as shown in Figure 6.1.

6.3 Identifying a Behavior Schedule with No Intermediate Goals

The problem in Equation (6.11) is quite challenging to approach directly, so we begin with a relaxed version of the problem where there are no intermediate goals (i.e. $\mathcal{Y} = \varnothing$). In this section, we begin by analyzing the simplest case, where the behavior library only contains two behaviors ($m = 2$) and we show how to find the optimal behavior schedule to achieve the single goal $\mathbf{x}_{\text{final}}$. We then extend our analysis to the case where there are multiple behaviors in the library.

6.3.1 Behavior Library Contains Only Two Behaviors

When the behavior library only contains two behaviors, any behavior schedule must necessarily alternate between selecting the two behaviors, so the only variables are the durations. Thus, the state of the system evolves as follows.

$$\dot{\mathbf{x}}(t) = \begin{cases} -\mathbf{L}(\mathbf{x}(t) - \mathbf{z}_1) & \text{for } t \in [0, t_1) \\ -\mathbf{L}(\mathbf{x}(t) - \mathbf{z}_2) & \text{for } t \in [t_1, t_2) \\ -\mathbf{L}(\mathbf{x}(t) - \mathbf{z}_1) & \text{for } t \in [t_2, t_3) \\ -\mathbf{L}(\mathbf{x}(t) - \mathbf{z}_2) & \text{for } t \in [t_3, t_4) \\ \quad \vdots \end{cases} \tag{6.12}$$

By considering the change of variables $\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{z}_1$, we can always analyze the equivalent system

$$\dot{\bar{\mathbf{x}}}(t) = \begin{cases} -\mathbf{L}\bar{\mathbf{x}}(t) & \text{for } t \in [0, t_1) \\ -\mathbf{L}(\bar{\mathbf{x}}(t) + \mathbf{z}_1 - \mathbf{z}_2) & \text{for } t \in [t_1, t_2) \\ -\mathbf{L}\bar{\mathbf{x}}(t) & \text{for } t \in [t_2, t_3) \\ -\mathbf{L}(\bar{\mathbf{x}}(t) + \mathbf{z}_1 - \mathbf{z}_2) & \text{for } t \in [t_3, t_4) \\ \quad \vdots \end{cases} \tag{6.13}$$

and find an optimal behavior schedule to transition from initial state $\bar{\mathbf{x}}_{\text{initial}} = \mathbf{x}_{\text{initial}} - \mathbf{z}_1$ to final state $\bar{\mathbf{x}}_{\text{final}} = \mathbf{x}_{\text{final}} - \mathbf{z}_1$. Thus, we will drop the overbar notation and analyze the following system.

$$\dot{\mathbf{x}}(t) = \begin{cases} -\mathbf{L}\mathbf{x}(t) & \text{for } t \in [0, t_1) \\ -\mathbf{L}(\mathbf{x}(t) - \mathbf{z}) & \text{for } t \in [t_1, t_2) \\ -\mathbf{L}\mathbf{x}(t) & \text{for } t \in [t_2, t_3) \\ -\mathbf{L}(\mathbf{x}(t) - \mathbf{z}) & \text{for } t \in [t_3, t_4) \\ \quad \vdots \end{cases} \tag{6.14}$$

Given the system above, the following important question naturally arises: what is the maximum number of switches required to achieve a goal $\mathbf{x}_{\text{final}}$? We will now seek to answer this question by considering the following problem that occurs in traditional time-optimal control of LTI systems with bounded control signals $\mathbf{u}(t) \in [0, 1]$.

Theorem 6. *The solution (if it exists) to the problem in Equation* (6.15) *is the optimal control signal* $\mathbf{u}^*(t)$*. This control signal corresponds to the optimal behavior schedule for the problem*

in Equation (6.11) *when* $\mathcal{Y} = \varnothing$ *and the state evolution corresponds to Equation* (6.14).

$$\begin{aligned} \underset{\mathbf{u}(t)}{\arg\max} \quad & \int_0^T -1 \; dt \\ \textit{subject to} \quad & \mathbf{x}(0) = \mathbf{x}_{\textit{initial}} \\ & \left\| \mathbf{P}\left(\mathbf{x}(T) - \mathbf{x}_{\textit{final}}\right) \right\|_2^2 \leq \varepsilon \\ & \dot{\mathbf{x}}(t) = -\mathbf{L}\mathbf{x}(t) + \mathbf{L}\mathbf{z}\mathbf{u}(t) \\ & \mathbf{u}(t) \in [0,1] \end{aligned} \tag{6.15}$$

Proof. The payoff functional $\int_0^T -1 \; dt = -T$ is maximized when the time T at which the swarm achieves the desired goal $\mathbf{x}_{\text{final}}$ is minimized. First, we form the Hamiltonian, where $\mathbf{p}(t)$ is the costate. Due to space constraints, we will drop the dependence on time t from our notation.

$$H(\mathbf{x}, \mathbf{p}, \mathbf{u}) = \mathbf{p}^\top \left(-\mathbf{L}\mathbf{x} + \mathbf{L}\mathbf{z}\mathbf{u}\right) - 1 \tag{6.16}$$

This is a free-time, constrained-endpoint problem, for which Pontryagin's Maximum Principle states the following conditions under which a given input signal $\mathbf{u}(t)$ is optimal.

$$\dot{\mathbf{x}} = \nabla_p H = -\mathbf{L}\mathbf{x} + \mathbf{L}\mathbf{z}\mathbf{u} \tag{6.17}$$

$$\dot{\mathbf{p}} = -\nabla_x H = \mathbf{L}^\top \mathbf{p} \tag{6.18}$$

$$\forall t \in [0,T] : H(\mathbf{x}, \mathbf{p}, \mathbf{u}) = \max_{a \in [0,1]} H(\mathbf{x}, \mathbf{p}, a) \tag{6.19}$$

$$\mathbf{p}(T) = \nabla_x \left(\left\| \mathbf{P}\left(\mathbf{x} - \mathbf{x}_{\text{final}}\right) \right\|_2^2 - \varepsilon \right) \Big|_{t=T} \tag{6.20}$$

Here, we have two differential equations representing the state $\mathbf{x}(t)$ and costate $\mathbf{p}(t)$ evolution. In addition, we have a terminal condition on the costate trajectory — it must be perpendicular to the curve representing the set of possible final states $\mathbf{x}(T)$ for the system. By expanding the third condition, we find

$$\begin{aligned} H(\mathbf{x}, \mathbf{p}, \mathbf{u}) &= \max_{a \in [0,1]} H(\mathbf{x}, \mathbf{p}, a) \\ \mathbf{u} &= \underset{a \in [0,1]}{\arg\max} \left(\mathbf{p}^\top \left(-\mathbf{L}\mathbf{x} + \mathbf{L}\mathbf{z}a\right) - 1 \right) \\ \mathbf{u} &= \underset{a \in [0,1]}{\arg\max} \left(\mathbf{p}^\top \mathbf{L}\mathbf{z}a \right) \end{aligned} \tag{6.21}$$

which implies the following for optimal $\mathbf{u}^*$.

$$\forall a \in [0,1] : \mathbf{p}^\top \mathbf{L}\mathbf{z}\left(\mathbf{u}^* - a\right) \geq 0 \tag{6.22}$$

$$\mathbf{u}^* = \begin{cases} 1 & \text{for } \mathbf{p}^\top \mathbf{L}\mathbf{z} \geq 0 \\ 0 & \text{for } \mathbf{p}^\top \mathbf{L}\mathbf{z} < 0 \end{cases} \tag{6.23}$$

Clearly, the above form of piecewise continuous $\mathbf{u}^*$ has an exactly corresponding representation as a discrete behavior schedule S^* and applying $\mathbf{u}^*$ to solve the problem in Equation (6.15) causes the state to evolve according to Equation (6.14) as required. □

Returning to our robotic swarm, Theorem 6 implies that even when the the supervisory operator is permitted to select a convex combination of the consensus-based behaviors in the library, the optimal behavior schedule only switches discretely between individual behaviors in the library.

Theorem 7. *The maximum length of the optimal behavior schedule S^* for a robotic swarm performing consensus-based behaviors with $\mathcal{Y} = \varnothing$ and with a behavior library containing $|\mathcal{B}| = 2$ behaviors is $|S^*| \leq n$.*

Proof. By expanding the costate evolution we find $\mathbf{p}^\top \mathbf{L}\mathbf{z} = \mathbf{p}\left(0\right)^\top e^{\mathbf{L}t}\mathbf{L}\mathbf{z}$, which implies the well known result that for time-optimal control of a linear time-invariant system where the control signal is constrained to a convex set, each component of the control signal only takes on values at its limits (in this case, 0 and 1) — that is, the signal is bang-bang — and that the optimal control signal only switches a finite number of times. Since the dynamics matrix for our consensus-based behavior is the Laplacian of an undirected communication graph, it is symmetric positive-definite with real entries and has an eigendecomposition $\mathbf{L} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$.

$$\begin{aligned}
&\mathbf{p}^\top \mathbf{L}\mathbf{z} \\
=&\mathbf{p}\left(0\right)^\top e^{\mathbf{L}t}\mathbf{L}\mathbf{z} \\
=&\mathbf{p}\left(0\right)^\top e^{\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top t}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top\mathbf{z} \\
=&\mathbf{p}\left(0\right)^\top \mathbf{Q}e^{\mathbf{\Lambda}t}\mathbf{Q}^\top\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top\mathbf{z} \\
=&\tilde{\mathbf{p}}\left(0\right)^\top e^{\mathbf{\Lambda}t}\mathbf{\Lambda}\tilde{\mathbf{z}} \\
=&\sum_i \tilde{p}_i\left(0\right)\tilde{z}_i\lambda_i e^{\lambda_i t}
\end{aligned} \tag{6.24}$$

Here $\tilde{\mathbf{z}} = \mathbf{Q}^\top\mathbf{z}$ and $\tilde{\mathbf{p}} = \mathbf{Q}^\top\mathbf{p}$. Since all the eigenvalues λ_i of $\mathbf{L}$ must be real and nonnegative, this is simply a sum of growing exponentials. Some of the coefficients may be positive and some of the coefficients may be negative, so this switches between positive and negative a maximum of n times over $t \in [0, T]$, which means that the optimal control signal must switch between 0 and 1 at most n times. This implies that the optimal behavior schedule for our swarm has a maximum length of n. □

We note that while the proofs of Theorem 6 and Theorem 7 were presented for a symmetric dynamics matrix for ease of exposition, they can actually be extended to any diagonalizable dynamics matrix in a straightforward manner.

6.3.2 Behavior Library Contains Many Behaviors

We now turn our attention to the problem of finding the optimal behavior schedule when our behavior library can contain any number of consensus-based behaviors and there are no intermediate goals (i.e. $\mathcal{Y} = \varnothing$). Rather than a vector $\mathbf{z} \in \mathbb{R}^n$, now consider a matrix $\mathbf{Z} \in \mathbb{R}^{n \times m}$ where the i-th column is the i-th bias vector $\mathbf{z}_i$ from the set of bias vectors $\mathcal{Z}$ corresponding to behaviors in the library $\mathcal{B}$.

Theorem 8. *For the problem in Equation* (6.15)*, if the dynamics constraint is replaced with* $\dot{\mathbf{x}}(t) = -\mathbf{L}\mathbf{x}(t) + \mathbf{L}\mathbf{Z}\mathbf{u}(t)$ *and* $\mathbf{u}(t)$ *is constrained to a simplex in the positive orthant (i.e.* $\mathbf{u}(t) \in [0,1]^m : \|\mathbf{u}(t)\|_1 \leq 1$*), then the optimal signal* $\mathbf{u}^*(t)$ *corresponds to the optimal behavior schedule for the problem in Equation* (6.11) *when* $\mathcal{Y} = \varnothing$ *and the state evolution corresponds to Equation* (6.8)*.*

Proof. Applying Pontryagin's Maximum Principle with $\mathcal{A} = \{\mathbf{a} \mid \mathbf{a} \in [0,1]^m : \|\mathbf{a}\|_1 \leq 1\}$, we find the following condition.

$$\begin{aligned} H(\mathbf{x}, \mathbf{p}, \mathbf{u}) &= \max_{\mathbf{a} \in \mathcal{A}} H(\mathbf{x}, \mathbf{p}, \mathbf{a}) \\ \mathbf{u} &= \arg\max_{\mathbf{a} \in \mathcal{A}} \left(\mathbf{p}^\top (-\mathbf{L}\mathbf{x} + \mathbf{L}\mathbf{Z}\mathbf{a}) - 1\right) \\ \mathbf{u} &= \arg\max_{\mathbf{a} \in \mathcal{A}} \left(\mathbf{p}^\top \mathbf{L}\mathbf{Z}\mathbf{a}\right) \end{aligned} \tag{6.25}$$

Expanding the costate evolution with $\mathbf{L} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$, we find

$$\mathbf{p}(t)^\top \mathbf{L}\mathbf{Z}\mathbf{a} = \sum_j a_j \sum_i \tilde{p}_i(0)\, \tilde{z}_{ij} \lambda_i e^{\lambda_i t} \tag{6.26}$$

with $\tilde{\mathbf{Z}} = \mathbf{Q}^\top \mathbf{Z}$ and $\tilde{\mathbf{p}} = \mathbf{Q}^\top \mathbf{p}$ which implies the following for optimal $\mathbf{u}^*$.

$$j^* = \arg\max_j \left(\sum_i \tilde{p}_i(0)\, \tilde{z}_{ij} \lambda_i e^{\lambda_i t} \right) \tag{6.27}$$

$$\forall j \neq j^* : u_j^*(t) = 0 \tag{6.28}$$

$$u_{j^*}^*(t) = \begin{cases} 1 & \text{for } \sum_i \tilde{p}_i(0)\, \tilde{z}_{ij^*} \lambda_i e^{\lambda_i t} \geq 0 \\ 0 & \text{for } \sum_i \tilde{p}_i(0)\, \tilde{z}_{ij^*} \lambda_i e^{\lambda_i t} < 0 \end{cases} \tag{6.29}$$

Thus, the optimal signal $\mathbf{u}^*(t)$ has either all entries equal to 0 or a single entry equal to 1 at any given time t. Clearly, it is piecewise continuous and has an exactly corresponding representation as an optimal behavior schedule S^* to solve Equation (6.11) when $\mathcal{Y} = \varnothing$. □

Theorem 8 implies that even when the supervisory operator is allowed to continuously switch between a convex combination of the behaviors in the library, the optimal behavior schedule marvelously only switches discretely between individual behaviors rather than selecting a weighted combination.

Theorem 9. *The maximum length of the optimal behavior schedule S^* for a robotic swarm performing consensus-based behaviors with $\mathcal{Y} = \varnothing$ and with a behavior library containing $|\mathcal{B}| = m$ behaviors is $|S^*| \leq \frac{m(m-1)}{2}n$.*

Proof. The proof is immediate from Theorem 8 since we may switch between any pair of behaviors in the library at most n times and there are $\frac{m(m-1)}{2}$ unique pairs. □

In practice, we expect the majority of behavior schedules to be much shorter than the maximum suggested by Theorem 9.

6.3.3 Procedure for Computing Behavior Schedule with No Intermediate Goals

Now that we have characterized the maximum length of the optimal behavior schedule, we present a procedure to compute it when $\mathcal{Y} = \varnothing$. Begin by assuming the length of the behavior schedule S we are optimizing is $d = \frac{m^2(m-1)}{2}n$ with all m behaviors in the library repeated $\frac{m(m-1)}{2}n$ times (i.e. $b_k = b_{mj+k}$ for $j = 0, 1, \ldots, \frac{m(m-1)}{2}n - 1$). The reason we select this length for the behavior schedule is that we can also use it to represent all possible sequences of behaviors of length $\frac{m(m-1)}{2}n$ (i.e. the maximum length of the true optimal behavior sequence) by simply deleting $\frac{m(m-1)^2}{2}n$ entries in the schedule (i.e. setting their corresponding durations to 0). This allows us to somewhat indirectly address the sequencing problem, which is combinatorial and computationally difficult to tackle directly.

Now we simply have to compute the corresponding durations, which we can represent as a vector $\bar{\tau} = \begin{bmatrix} \tau_1 & \tau_2 & \ldots & \tau_d \end{bmatrix}^\top$. Since all of our behaviors are consensus-based behaviors, we can write the state evolution recursively.

$$\mathbf{x}\left(t_k\right) = e^{-\mathbf{L}\tau_k}\left(\mathbf{x}\left(t_{k-1}\right) - \mathbf{z}_{b_k}\right) + \mathbf{z}_{b_k} \tag{6.30}$$

Expanding and simplifying, we get the following closed form expression for the state evolution.

$$\mathbf{x}\left(t_d\right) = e^{-\mathbf{L}\sum_{i=1}^{d}\tau_i}\mathbf{x}\left(0\right) - \sum_{j=1}^{d} e^{-\mathbf{L}\sum_{i=j}^{d}\tau_i}\mathbf{z}_{b_j} + \sum_{j=1}^{d-1} e^{-\mathbf{L}\sum_{i=j+1}^{d}\tau_i}\mathbf{z}_{b_j} + \mathbf{z}_{b_d} \tag{6.31}$$

We can now rewrite the problem in Equation (6.11) as follows (assuming $\mathcal{Y} = \varnothing$) and noting that $T = t_d$.

$$\begin{aligned} \underset{\bar{\tau} \in \mathbb{R}^d}{\arg\min} \quad & \mathbf{1}^\top \bar{\tau} \\ \text{subject to} \quad & \mathbf{x}(0) = \mathbf{x}_{\text{initial}} \\ & \left\| \mathbf{P}\left(\mathbf{x}(t_d) - \mathbf{x}_{\text{final}}\right) \right\|_2^2 \leq \varepsilon \\ & \mathbf{x}(t_d) = e^{-\mathbf{L}\sum_{i=1}^{d}\tau_i}\mathbf{x}(0) - \sum_{j=1}^{d} e^{-\mathbf{L}\sum_{i=j}^{d}\tau_i}\mathbf{z}_{b_j} + \sum_{j=1}^{d-1} e^{-\mathbf{L}\sum_{i=j+1}^{d}\tau_i}\mathbf{z}_{b_j} + \mathbf{z}_{b_d} \end{aligned} \tag{6.32}$$

This is a nonlinear optimization problem with a single equality constraint and nonnegative optimization variables. Conveniently, in this form, both the objective function and the constraints are continuous and smooth in $\bar{\tau}$. The gradient of the objective function is given as follows.

$$\nabla_{\bar{\tau}} T = \mathbf{1} \tag{6.33}$$

The gradient of the equality constraint is given as follows.

$$\begin{aligned} \phi(\mathbf{x}) &= \left\| \mathbf{P}\left(\mathbf{x}(T) - \mathbf{x}_{\text{final}}\right) \right\|_2^2 - \varepsilon \\ \nabla_{\bar{\tau}} \phi &= 2 \begin{bmatrix} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau_1} & \dots & \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau_d} \end{bmatrix}^\top \mathbf{P}^\top \mathbf{P}\left(\mathbf{x}(T) - \mathbf{x}_{\text{final}}\right) \end{aligned} \tag{6.34}$$

The derivative of the state evolution with respect to each duration is given as follows.

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau_k} = -\mathbf{L} e^{-\mathbf{L}\sum_{i=1}^{d}\tau_i}\mathbf{x}(0) - \sum_{j=1}^{k} -\mathbf{L} e^{-\mathbf{L}\sum_{i=j}^{d}\tau_i}\mathbf{z}_{b_j} + \sum_{j=1}^{k-1} -\mathbf{L} e^{-\mathbf{L}\sum_{i=j+1}^{d}\tau_i}\mathbf{z}_{b_j} \tag{6.35}$$

Since we have analytic gradients for both the objective function and the constraints, we can use any number of standard gradient-based optimization techniques to find locally optimal solutions efficiently (see Section 6.5 for an example). As noted above, we expect the vast majority of the durations in our behavior schedule to be 0 because we intentionally increased its length to enable it to represent all possible behavior sequences. In addition, if the supervisory operator for the robotic swarm knows the behavior sequence in advance, our technique can be used to identify the optimal durations for which those behaviors should be executed.

Finally, we note that our technique applies equally well to a general dynamics matrix $\mathbf{A}$ and it is obvious that computational efficiency can be improved when $\mathbf{A}$ is diagonalizable by taking the eigendecomposition $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$ and considering transformed state $\tilde{\mathbf{x}}(t) = \mathbf{Q}^{-1}\mathbf{x}(t)$, and behaviors $\tilde{f}_j(\tilde{\mathbf{x}}(t)) = \mathbf{\Lambda}(\tilde{\mathbf{x}}(t) - \tilde{\mathbf{z}}_j)$ with $\tilde{\mathbf{z}}_j = \mathbf{Q}^{-1}\mathbf{z}_j$. For these transformed behaviors, the dynamics matrix $\mathbf{\Lambda}$ is diagonal, which makes its matrix exponential $e^{\mathbf{\Lambda} t}$ much faster to compute. Clearly, this transformation is for computational purposes only and does not change either the mathematical results or the physical motion of robots in our swarm.

6.4 Algorithm for Sequencing of Unordered Intermediate Goals

In the previous section, we showed that when there are no intermediate goals, it is possible to identify the maximum length of the time-optimal behavior schedule, fix the sequence and then solve a nonlinear optimization problem with smooth objective function and smooth constraints for the durations. The identified behavior schedule will be locally optimal in the space of durations. Given a particular behavior library, we can do this for any choice of initial state and goal.

We now consider the problem of behavior scheduling with a set of c intermediate goals $\mathcal{Y}$ that must be achieved by the robotic swarm. Consider the weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$. The vertices $\mathcal{V} = \mathcal{Y} \cup \{\mathbf{x}_{\text{initial}}, \mathbf{x}_{\text{final}}\}$ of this graph include the initial state, the intermediate goals and the final goal. The edge set $\mathcal{E}$ includes directed edges between every pair of nodes but does not include any edges that begin at $\mathbf{x}_{\text{final}}$ or end at $\mathbf{x}_{\text{initial}}$. The entries $w_{ij} \in \mathbb{R}_+$ of the weight matrix $\mathbf{W} \in \mathbb{R}^{(c+2)\times(c+2)}$ represent the minimum time required to transition from the state represented by vertex v_i to the state represented by vertex v_j. These weights may be computed by using the procedure in the previous section to find the optimal behavior schedule to move from v_i to v_j. Thus, we note that weighted edges in our directed graph must satisfy the triangle inequality (i.e. $\forall i, j, k : w_{ij} + w_{jk} \geq w_{ik}$). In the rare case that particular triplets do not satisfy the triangle inequality (since our procedure is only locally optimal), we enforce the triangle inequality by replacing the computed behavior schedule for edge (v_i, v_k) with the concatenation of the computed behavior schedules for (v_i, v_j) and (v_j, v_k).

Once we have created the weight matrix $\mathbf{W}$, which is not symmetric for our problem, and computed the behavior schedules, our problem is reduced to finding the minimum cost Hamiltonian path which starts at $\mathbf{x}_{\text{initial}}$ and ends at $\mathbf{x}_{\text{final}}$. This is a variation of the Metric Asymmetric Travelling Salesman Problem (TSP), which is known to be NP-hard. Christofides' approximation algorithm solves the Metric Symmetric TSP problem in O($|\mathcal{V}|^3$) with bounded suboptimality [21]. Specifically, the cost of the tour produced by Christofides' algorithm will be at most $\frac{3}{2}$ times the cost of the optimal tour. In [47], the authors show that the Asymmetric TSP problem can be transformed into the Symmetric TSP, problem after which we can apply Christofides' algorithm as usual to find a $\frac{3}{2}$-approximation to the optimal tour. However, our problem also has fixed endpoints, so we must apply a variant of Christofides' algorithm [39] for Hamiltonian paths rather than tours. It is shown in [39] that when there are two fixed endpoints, the algorithm produces a $\frac{5}{3}$-approximation to the optimal Hamiltonian path. The behavior schedule S^* which solves the problem in Equation (6.11) is the concatenation of the locally optimal behavior schedules for each edge in the Hamiltonian path from initial state $\mathbf{x}_{\text{initial}}$ through intermediate goals $\mathcal{Y}$ to final

Algorithm 2 Compute the Best Behavior Schedule

1: **function** MULTIGOALBEHAVIORSCHEDULE($\mathcal{B}, \mathbf{x}_{\text{initial}}, \mathcal{Y}, \mathbf{x}_{\text{final}}$)
2: $S_{\text{all}} \leftarrow \varnothing$, $\mathcal{V} = \mathcal{Y} \cup \{\mathbf{x}_{\text{initial}}, \mathbf{x}_{\text{final}}\}$
3: **for** $v_i \in \mathcal{V} : v_i \neq \mathbf{x}_{\text{final}}$ **do**
4: **for** $v_j \in \mathcal{V} : (v_i \neq v_j) \wedge (v_j \neq \mathbf{x}_{\text{initial}})$ **do**
5: $S_{ij} \leftarrow$ ONEGOALBEHAVIORSCHEDULE($\mathcal{B}, v_i, v_j$)
6: $w_{ij} \leftarrow$ TOTALDURATION(S_{ij})
7: **end for**
8: **end for**
9: $\mathbf{W} \leftarrow$ ENFORCETRIANGLEINEQUALITY($\mathbf{W}$)
10: $P \leftarrow$ FIXEDENDPOINTSHAMILTONIANPATH($\mathbf{W}, \mathbf{x}_{\text{initial}}, \mathbf{x}_{\text{final}}$)
11: **for** $(v_i, v_j) \in P$ **do**
12: $S_{\text{all}} \leftarrow S_{\text{all}} \sqcup S_{ij}$
13: **end for**
14: **return** S_{all}
15: **end function**

goal $\mathbf{x}_{\text{final}}$. The behavior schedule S^* has bounded local suboptimality.

The method described above is summarized in Algorithm 2. The function ONEGOALBEHAVIORSCHEDULE() solves the problem in Equation (6.32). Notice that the locally optimal behavior schedule for every edge and every entry of the weight matrix can be computed in parallel (lines 3–8).

We conclude this section by noting that if the final goal is not fixed (i.e. no $\mathbf{x}_{\text{final}}$), then we can use the algorithm in [39] to find a single endpoint Hamiltonian path starting at $\mathbf{x}_{\text{initial}}$ and achieving all $\mathcal{Y}$ with $\frac{3}{2}$ bounded suboptimality. This fact can also be used to extend Algorithm 2 in a straightforward manner to handle ordered subsets of unordered goals (e.g. achieve all goals in $\mathcal{Y}_1$, then $\mathcal{Y}_2$, then $\mathcal{Y}_3$, etc.).

6.5 Application to Configuration Control of Robotic Swarms

In this section, we revisit the problem of optimal timing in configuration control of robotic swarms, which we presented to illustrate the phenomenon of Neglect Benevolence in [57]. There are many applications, including artistic performances [17, 73], surveillance, cooperative manipulation or robotic printing where it is important to generate and maintain various configurations with a multi-robot team. In some of these applications, the sequence of desired configurations is known in advance and in others the desired configurations are known, but not the sequence. Given a library of consensus-based behaviors $\mathcal{B}$ that each cause the swarm to move towards a particular spatial configuration $\mathbf{z} \in \mathcal{Z}$, we now apply the results from the previous sections to

identify a behavior schedule S that generates the desired configurations $\mathcal{Y}$ in minimum time. For these examples, we make the reasonable assumption that $\mathcal{Y} \subset \mathcal{Z}$. The optimal behavior schedule S will *transiently* switch between configurations $\mathcal{Z}$ in our library to ensure that the swarm achieves (to ε-convergence) all unordered goals $\mathcal{Y}$ and final goal $\mathbf{x}_{\text{final}}$ in minimum time. We instantiate our behavior library $\mathcal{B}$ with a set of configurations $\mathcal{Z}$ shown in Figure 6.2.

6.5.1 Known Behavior Sequence with Unknown Durations

We first consider a case similar to the one we considered in [57]. The human operator is performing a mission with the robotic swarm and has applied behavior f_2, which causes the swarm to move towards the corresponding configuration $\mathbf{z}_2$. However, at time $t = t_0 = 0$ when the swarm is in state $\mathbf{x}_{\text{initial}}$ (random configuration), the operator becomes aware of a change in mission goal to $\mathbf{x}_{\text{final}}$, which can be achieved using behavior f_3 (e.g. $\|\mathbf{P}(\mathbf{z}_3 - \mathbf{x}_{\text{final}})\|_2^2 = 0$). Based on this description, the behavior schedule must have the form $S = ((2, \tau_1), (3, \tau_2))$. Note that $\mathbf{x}_{\text{initial}}$ is Neglect Benevolent simply if the optimal schedule has $\tau_1 > 0$. In our simulation (which had different $\mathbf{x}_{\text{initial}}$ than [57]) when $|S| = 2$, the schedule computed is $S =$ ((2,12.5267), (3,38.0237)) with total time $T = 50.55$. When we allow $|S| = n$, which is the maximum possible entries in the optimal schedule, the non-zero duration entries in the schedule computed are $S =$ ((3,19.2606), (2,2.9868), (3,27.9326)) with total time $T = 50.18$, a slight improvement.

6.5.2 Target Configuration with No Desired Intermediate Configurations

We now consider the case where we don't know the behavior sequence in advance, but want to find the behavior schedule to achieve $\mathbf{x}_{\text{final}} = \mathbf{z}_7$ from $\mathbf{x}_{\text{initial}}$ (random configuration) in minimum time. We implemented the procedure in Section 6.3.3 in MATLAB and used `fmincon` (with `'interior-point'` and analytic gradients specified) to identify the optimal behavior schedule $S =$ ((5,0.1198), (6,0.8556), (8,0.1277), (5,15.9487), (8,0.4150), (5,2.1281), (7,35.4172)) for a total duration $T = 55.01$. We only show entries with non-zero duration, but note that the length of the behavior schedule is significantly less than the maximum suggested in Theorem 9. Clearly, only two of the behaviors (5 and 7) were applied for a significant portion of time.

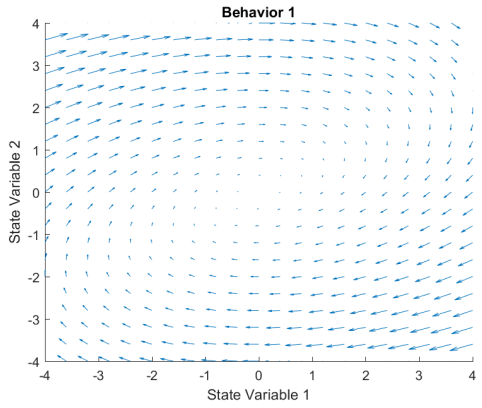
6.5.3 Target Configuration with Desired Intermediate Configurations

Finally, we consider the case where we want to find a behavior schedule that achieves a set of intermediate configurations $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2\}$ (with $\mathbf{y}_1 = \mathbf{z}_2$ and $\mathbf{y}_2 = \mathbf{z}_3$) from $\mathbf{x}_{\text{initial}}$ (random con-

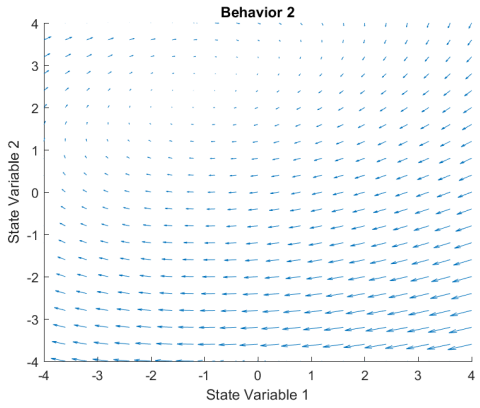
figuration) to $\mathbf{x}_{\text{final}} = \mathbf{z}_7$ in minimum time. We apply the algorithm in Section 6.4 to our problem and identify the optimal behavior schedule $S =$ ((3,1.3048), (4,3.7551), (5,0.0914), (3,10.8722), (5,6.7721), (3,31.0313), (8,1.0880), (2,11.7012), (7,1.0925), (2,20.9164), (7,0.4393), (7,0.8738), (7,23.8005)) for a total duration $T = 113.74$. We only show entries with non-zero duration, but note that the length of the behavior schedule is significantly less than the maximum suggested in Theorem 9. Intermediate goal $\mathbf{y}_2 = \mathbf{z}_3$ was reached at $t = 53.83$ and goal $\mathbf{y}_1 = \mathbf{z}_2$ was reached at $t = 89.06$.

6.6 Conclusion

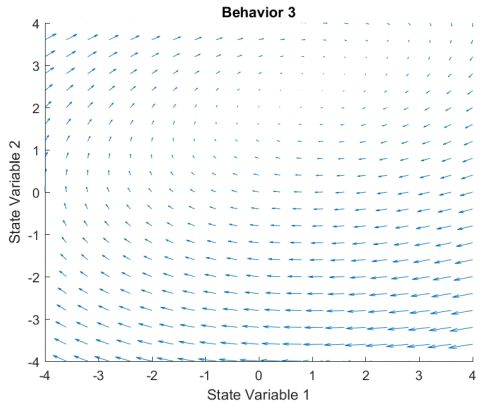
In this chapter, given a library of consensus-based behaviors, we considered the problem of generating a behavior schedule that enables a robotic swarm to achieve a set of unordered intermediate goals and a final goal in minimum time. After identifying a procedure to produce a locally optimal behavior schedule between any two goals, we presented an algorithm that computes locally optimal behavior schedules between all pairs of goals and then applies a variant of Christofides' algorithm to find a $\frac{5}{3}$-approximation to the optimal Hamiltonian path through the goals. Concatenating the behavior schedules along path, we obtained an overall behavior schedule. In this work, our only cost was on time to achieve all goals. In future work, we plan to explore other cost functions and more types of behaviors in our library.



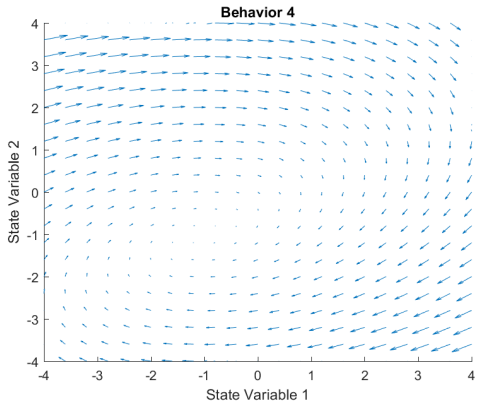
(a) Behavior 1 Defined by Bias $\mathbf{z}_1$



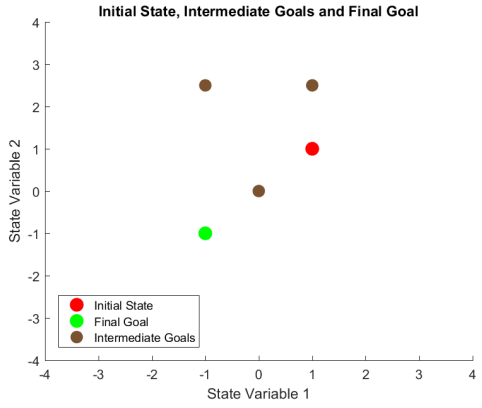
(b) Behavior 2 Defined by Bias $\mathbf{z}_2$



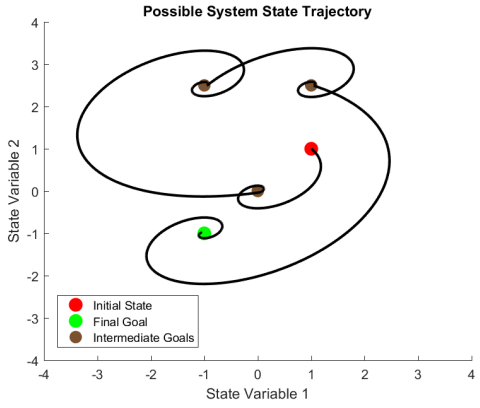
(c) Behavior 3 Defined by Bias $\mathbf{z}_3$



(d) Behavior 4 Defined by Bias $\mathbf{z}_4$

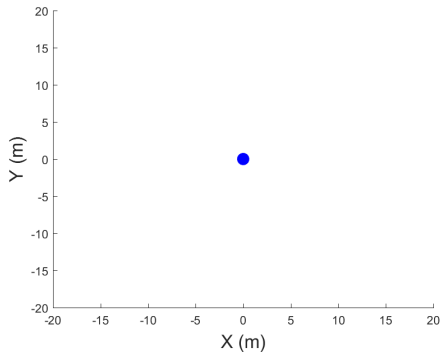


(e) Initial State (red), Final Goal (green) and Intermediate Goals (brown)

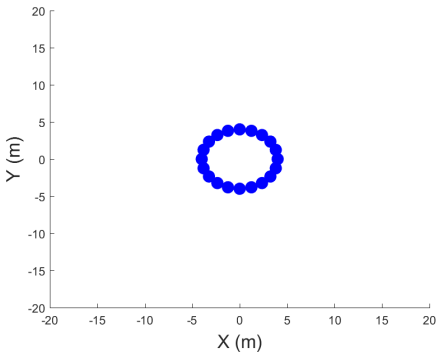


(f) Feasible Solution Trajectory that Achieves All Intermediate Goals and Final Goal

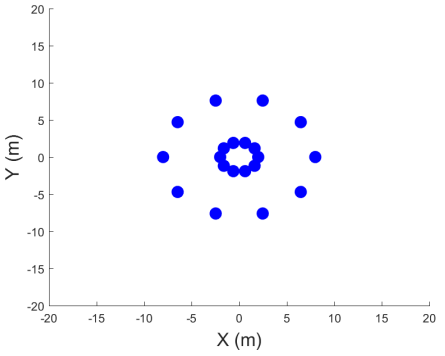
Figure 6.1: Given a library containing four behaviors with LTI dynamics defined by a common stable dynamics matrix $\mathbf{A}$ but different equilibrium points, we want to identify a behavior schedule (behavior sequence and corresponding durations of application) that minimizes the total time to achieve all unordered intermediate goals and the final goal.



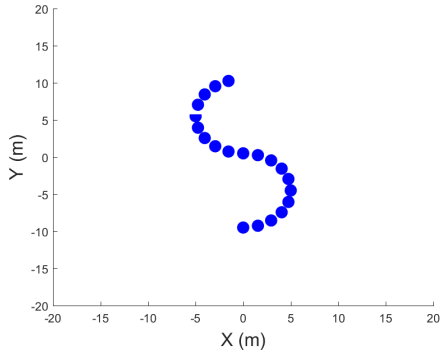
(a) Rendezvous ($\mathbf{z}_1$)



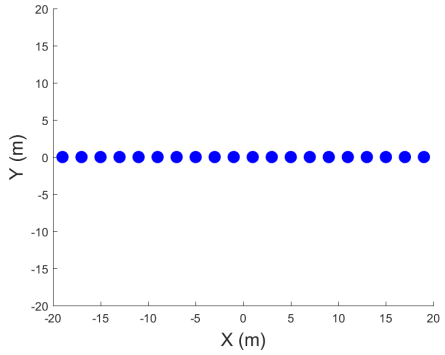
(b) Circle ($\mathbf{z}_2$)



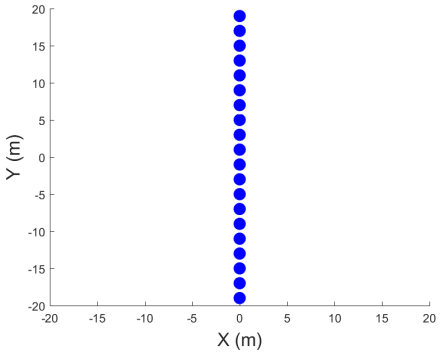
(c) Torus ($\mathbf{z}_3$)



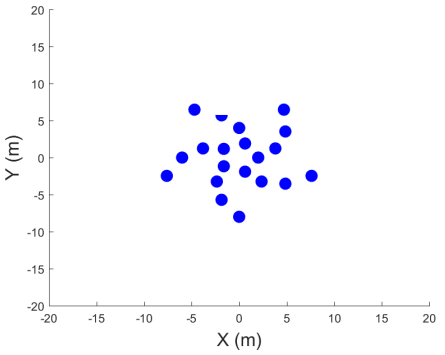
(d) S ($\mathbf{z}_4$)



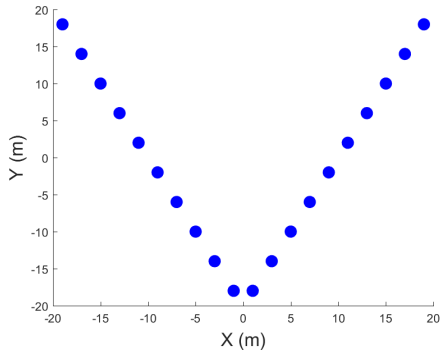
(e) Line X ($\mathbf{z}_5$)



(f) Line Y ($\mathbf{z}_6$)



(g) Spiral ($\mathbf{z}_7$)



(h) V ($\mathbf{z}_8$)

Figure 6.2: Equilibrium Configurations for Behaviors in Our Library

Chapter 7

Conclusion

7.1 Summary of Contributions

In this thesis, we studied the challenging problem of behavior composition in supervisory interaction with robotic swarms. Since swarms are large distributed dynamical systems, it is difficult to predict the effect of an input on the state of the swarm. Our contributions included a formalization of the swarm behavior composition problem, characterizing the impact of input timing on swarm performance and algorithms to optimally schedule swarm behaviors to optimize a performance criterion.

Our approach began with a principled study of the impact of input timing on human-swarm interaction and then applying these results to inform the development of methods for automated composition of swarm behaviors from a library to accomplish tasks.

We introduced the Neglect Benevolence phenomenon, the idea that delaying an input could sometimes improve the performance of the swarm, and formalized it in a control theoretic framework. Interestingly, while we studied this phenomenon in the context of robotic swarms, these ideas are more universally applicable to distributed dynamical systems. Using a simple algorithm to identify the optimal input time, we applied it to swarm configuration control to find the best time to switch between two swarm behaviors where each behavior achieved a different configuration. Treating this problem of switching between two swarm behaviors achieving different configurations as a base human-swarm interaction task in which we could automatically identify optimal input times, we demonstrated with an empirical study that humans can learn to approximate optimal input times and that visual aids can be developed to augment human interfaces. We expect that these results will be very valuable in future design of augmentations for

human-swarm interaction interfaces.

We next considered the issue of timing in an adversarial setting involving a swarm of agents applying discrete-time consensus, but executing updates periodically with different frequencies and delays. In this situation, we showed that we could systematically influence the agreement point of the entire swarm by influencing the update periods and delays of only a subset of agents. We believe these results will be important to developing mitigation techniques in the future for various timing-only approaches to adversarial subversion of swarm performance.

We then tackled the problem of optimally sequencing swarm behaviors selected from a library of concrete behaviors to accomplish tasks for which no individual behavior had been designed. Our contribution was an algorithmic approach based on informed search that was proven to be optimal and complete. Our results demonstrated that the approach could be used for multiple applications, even those for which behaviors in the library had not been designed. Despite the assumption that the durations for each behavior are known, we believe this technique represents a significant step forward toward composing general swarm behaviors.

Finally, we solved the problem of finding the optimal behavior sequence and the associated switch times simultaneously in a process we referred to as behavior scheduling. This is a very challenging problem for general swarm behaviors, so we restricted ourselves to consensus-based swarm behaviors which are widely used in the literature and have representative applications such as rendezvous or configuration control. Our contribution was a reformulation that permitted finding analytical gradients for our problem, which could be used to identify a locally optimal solution in any gradient-based solver. We extended our approach to achieve multiple unordered goals with bounded suboptimality on the sequence in which we achieve the goals. We believe this approach represents an important step forward for swarm behavior composition since consensus is essential to many swarm behaviors.

7.2 Future Work

As is often the case with any research, there are always more unanswered questions. In this section, we outline a few directions for future work.

7.2.1 Instantiating a Library of Concrete Behaviors from Meta-Behaviors

In this thesis, we approached the swarm behavior composition problem from the perspective that our library of behaviors consisted of concrete behaviors with no unspecified parameters rather

than meta-behaviors with some unspecified parameters. However, in some cases, it may not be possible to achieve the task with the given concrete behaviors. In this case, we can ask a few questions:

1. If we cannot achieve our goals with the current set of concrete behaviors in our library, but we can instantiate new concrete behaviors into our library from a set of meta-behaviors, which new concrete behaviors should we instantiate?
2. How do we select the appropriate values for the parameters to the corresponding meta-behaviors in order to instantiate concrete behaviors with the desired properties?
3. What is the minimal set of concrete swarm behaviors that should be in our library to accomplish our desired goals?

Some of these questions can be answered in a straightforward manner for certain special cases. For example, for consensus-based behaviors, if a certain goal cannot be achieved with the current concrete behaviors in the library, then one can instantiate a new consensus-based concrete behavior where the bias vector is the currently unachievable goal. However, this approach quickly becomes unwieldy if applied successively to a series of goals explicitly crafted to force a new concrete behavior being instantiated every time. In addition, it will not necessarily result in a minimal library of concrete behaviors to achieve all goals.

These questions are very challenging to answer in the general case. Progress towards their answers is of great importance to future work on swarm behavior composition.

Bibliography

[1] Daron Acemoglu and Asuman Ozdaglar. Opinion dynamics and learning in social networks. *Dynamic Games and Applications*, 1(1):3–49, 2011. 4, 4.4.2

[2] Sandip Aine, Siddharth Swaminathan, Venkatraman Narayanan, Victor Hwang, and Maxim Likhachev. Multi-heuristic a*. *The International Journal of Robotics Research*, 35(1-3): 224–243, 2016. 5.3

[3] Hideki Ando, Yoshinobu Oasa, Ichiro Suzuki, and Masafumi Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999. 5.1

[4] Ronald C Arkin. *Behavior-based robotics*. MIT press, 1998. 5

[5] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *IEEE transactions on robotics and automation*, 14(6):926–939, 1998. 2.1, 3.1

[6] Tucker Balch and Maria Hybinette. Social potentials for scalable multi-robot formations. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 73–80. IEEE, 2000. 5.1

[7] Shishir Bashyal and Ganesh Kumar Venayagamoorthy. Human swarm interaction for radiation source search and localization. In *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*, pages 1–8. IEEE, 2008. 2, 2.1, 5, 6

[8] Randal W Beard, Jonathan Lawton, Fred Y Hadaegh, et al. A coordination architecture for spacecraft formation control. *IEEE Transactions on control systems technology*, 9(6): 777–790, 2001. 2.1

[9] Calin Belta, Antonio Bicchi, Magnus Egerstedt, Emilio Frazzoli, Eric Klavins, and George J Pappas. Symbolic planning and control of robot motion [grand challenges of robotics]. *Robotics & Automation Magazine, IEEE*, 14(1):61–70, 2007. 5

[10] Bennett I Bertenthal and Jeannine Pinto. Global processing of biological motions. *Psychological science*, 5(4):221–225, 1994. 3.4

[11] Subhrajit Bhattacharya, Robert Ghrist, and Vijay Kumar. Multi-robot coverage and exploration on riemannian manifolds with boundaries. *The International Journal of Robotics Research*, 33(1):113–137, 2014. 6

[12] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013. 4, 5.1, 6

[13] Daniel S Brown, Sean C Kerman, and Michael A Goodrich. Human-swarm interactions based on managing attractors. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 90–97. ACM, 2014. 3.1, 3.4

[14] David J Bruemmer, Donald D Dudenhoeffer, Mark D McKay, and Matthew O Anderson. A robotic swarm for spill finding and perimeter formation. Technical report, DTIC Document, 2002. 2.1, 3.1

[15] Axel Buchner, Joachim Funke, and Dianne C Berry. Negative correlations between control performance and verbalizable knowledge: Indicators for implicit learning in process control tasks? *The Quarterly Journal of Experimental Psychology*, 48(1):166–187, 1995. 3

[16] Francesco Bullo, Jorge Cortes, and Sonia Martinez. *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009. 2.1, 3.1, 3.4

[17] Ellen A Cappo, Arjav Desai, and Nathan Michael. Robust coordinated aerial deployments for theatrical applications given online user interaction via behavior composition. In *Distributed Autonomous Robotic Systems*, pages 665–678. Springer, 2018. 6.5

[18] Yin Chen, Roberto Tron, Andreas Terzis, and Rene Vidal. Corrective consensus: Converging to the exact average. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 1221–1228. IEEE, 2010. 4

[19] Shih-Yi Chien, Michael Lewis, Siddharth Mehrotra, Nathan Brooks, and Katia Sycara. Scheduling operator attention for multi-robot control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 473–479. IEEE, 2012. 2.1

[20] Howie Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1-4):113–126, 2001. 2.1, 3.1

[21] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976. 6, 6.1, 6.4

[22] Gilles Coppin and François Legras. Autonomy spectrum and performance perception issues in swarm supervisory control. *Proceedings of the IEEE*, 100(3):590–603, 2012. 2.1, 3.1

[23] Jorge Cortés, Sonia Martínez, and Francesco Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, 51(8):1289–1298, 2006. 5.1

[24] Iain D Couzin, Jens Krause, Richard James, Graeme D Ruxton, and Nigel R Franks. Collective memory and spatial sorting in animal groups. *Journal of theoretical biology*, 218 (1):1–11, 2002. 2.1, 3.1

[25] Jacob W Crandall and Michael A Goodrich. Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1290–1295. IEEE, 2002. 2.1

[26] Lisa J Croner and Thomas D Albright. Image segmentation enhances discrimination of motion in visual noise. *Vision research*, 37(11):1415–1427, 1997. 3.4

[27] M Cummings. Human supervisory control of swarming networks. In *2nd Annual Swarming: Autonomous Intelligent Networked Systems Conference*, pages 1–9. Citeseer, 2004. 2.1

[28] James E Cutting, Cassandra Moore, and Roger Morrison. Masking the motions of human gait. *Perception & Psychophysics*, 44(4):339–347, 1988. 3.4

[29] Jean-Pierre de la Croix and Magnus Egerstedt. Controllability characterizations of leader-based swarm interactions. In *AAAI Fall Symposium: Human Control of Bioinspired Swarms*, 2012. 3.1

[30] Frederick Ducatelle, Gianni A Di Caro, and Luca M Gambardella. Cooperative self-organization in a heterogeneous swarm robotic system. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 87–94. ACM, 2010. 2.1, 3.1

[31] M Egerstedt and Xiaoming Hu. Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, 17(6):947–951, 2001. 2.1, 3.1

[32] Magnus Egerstedt, Yorai Wardi, and Florent Delmotte. Optimal control of switching times in switched dynamical systems. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 3, pages 2138–2143. IEEE, 2003. 6.1

[33] Lei Fang and Panos J Antsaklis. Information consensus of asynchronous discrete-time multi-agent systems. In *American Control Conference, 2005. Proceedings of the 2005*, pages 1883–1888. IEEE, 2005. 4, 4.1, 4.3.1

[34] MaryAnne Fields, Ellen Haas, Susan Hill, Christopher Stachowiak, and Laura Barnes. Effective robot team control methodologies for battlefield applications. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5862–5867. IEEE, 2009. 2.1, 3.1, 5, 6

[35] Gianpiero Francesca and Mauro Birattari. Automatic design of robot swarms: Achievements and challenges. *Frontiers in Robotics and AI*, 3(29), 2016. 6

[36] Veysel Gazi and Kevin M Passino. Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):539–557, 2004. 2, 3.2

[37] Michael A Goodrich, Brian Pendleton, PB Sujit, and José Pinto. Toward human interaction with bio-inspired robot teams. In *Systems, man, and cybernetics (smc), 2011 ieee international conference on*, pages 2859–2864. IEEE, 2011. 2.1

[38] ERFW Grossman and E Cooke. Manual control of slow-response systems. *Ergonomics: Major Writings*, page 281, 2005. 3

[39] JA Hoogeveen. Analysis of christofides' heuristic: Some paths are more difficult than cycles. *Operations Research Letters*, 10(5):291–295, 1991. 6, 6.1, 6.4, 6.4

[40] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & psychophysics*, 14(2):201–211, 1973. 3.4

[41] John G Kemeny, James Laurie Snell, et al. *Finite markov chains*, volume 356. van Nostrand Princeton, NJ, 1960. 4.2.1

[42] Zsolt Kira and Mitchell A Potter. Exerting human control over decentralized robot swarms. In *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*, pages 566–571. IEEE, 2009. 2.1, 3.1, 5, 6

[43] Günther Knoblich and Rüdiger Flach. Predicting the effects of actions: Interactions of perception and action. *Psychological Science*, 12(6):467–472, 2001. 3.4

[44] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis. Human interaction with robot swarms: A survey. *IEEE Transactions on Human-Machine Systems*, 46(1):9–26, Feb 2016. 5, 5.1, 6

[45] Andreas Kolling, Steven Nunnally, and Michael Lewis. Towards human control of robot swarms. In *Proceedings of the seventh annual ACM/IEEE international conference on human-robot interaction*, pages 89–96. ACM, 2012. 2, 2.1, 3.1, 3.8, 5, 6

[46] Andreas Kolling, Phillip Walker, Nilanjan Chakraborty, Katia Sycara, and Michael Lewis. Human interaction with robot swarms: A survey. *IEEE Transactions on Human-Machine Systems*, 46(1):9–26, 2016. 4

[47] Ratnesh Kumar and Haomin Li. On asymmetric tsp: Transformation to symmetric tsp and performance bound. *Journal of Operations Research*, 1996. 6.4

[48] Brian R Levinthal and Steven L Franconeri. Common-fate grouping as feature selection.

Psychological science, page 0956797611418346, 2011. 3.4

[49] Hai Lin and Panos J Antsaklis. Hybrid dynamical systems: An introduction to control and verification. *Foundations and Trends® in Systems and Control*, 1(1):1–172, 2014. 6, 6.1

[50] Peng Lin and Yingmin Jia. Consensus of second-order discrete-time multi-agent systems with nonuniform time-delays and dynamically changing topologies. *Automatica*, 45(9): 2154–2158, 2009. 4

[51] Yuri K Lopes, Stefan M Trenkwalder, André B Leal, Tony J Dodd, and Roderich Groß. Supervisory control theory applied to swarm robotics. *Swarm Intelligence*, 10(1):65–97, 2016. 6

[52] Sandra Mau and John M Dolan. Scheduling to minimize downtime in human-multirobot supervisory control. In *Proceedings of the 5th International Workshop on Planning and Scheduling for Space*. Citeseer, 2006. 2.1, 3.1

[53] James McLurkin, Jennifer Smith, James Frankel, David Sotkowitz, David Blau, and Brian Schmidt. Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 72–75, 2006. 3

[54] PM Mitchell, ML Cummings, and TB Sheridan. Management of multiple dynamic human supervisory control tasks. In *10th International Command and Control Research and Technology Symposium*, 2005. 2.1, 3.1

[55] Neville Moray, Pam Lootsteen, and Jan Pajak. Acquisition of process control skills. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(4):497–504, 1986. 3

[56] Nancy M Morris and William B Rouse. The effects of type of knowledge upon human problem solving in a process control task. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-15(6):698–707, 1985. 3

[57] Sasanka Nagavalli, Lingzhi Luo, Nilanjan Chakraborty, and Katia Sycara. Neglect benevolence in human control of robotic swarms. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6047–6053. IEEE, 2014. 1.4, 3.1, 3.2, 3.3, 3.3, 3.5.1, 4, 4.1, 5, 5.4.2, 6, 6.5, 6.5.1

[58] Sasanka Nagavalli, Shih-Yi Chien, Michael Lewis, Nilanjan Chakraborty, and Katia Sycara. Bounds of neglect benevolence in input timing for human interaction with robotic swarms. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 197–204. ACM, 2015. 1.4, 4, 5

[59] Sasanka Nagavalli, Nilanjan Chakraborty, and Katia Sycara. Automated sequencing of swarm behaviors for supervisory control of robotic swarms. In *Robotics and Automation*

(ICRA), 2017 IEEE International Conference on, pages 2674–2681. IEEE, 2017. 1.4, 6, 6.2.1

[60] Sasanka Nagavalli, Nilanjan Chakraborty, and Katia Sycara. On time-optimal behavior scheduling of robotic swarms for achieving multiple goals. In *2017 IEEE Conference on Automation Science and Engineering (CASE 2017)*, pages 1546–1553. IEEE, 2017. 1.4, 4

[61] Sasanka Nagavalli, Ramitha Sundar, and Katia Sycara. Exploiting asynchrony in multi-agent consensus to change the agreement point. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2018)*. IEEE, 2018. 1.4

[62] Amir M Naghsh, Jeremi Gancet, Andry Tanoto, and Chris Roast. Analysis and design of human-robot swarm interaction in firefighting. In *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pages 255–260. IEEE, 2008. 5, 6

[63] Reza Olfati-Saber and Richard M Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, 49(9):1520–1533, 2004. 4

[64] Reza Olfati-Saber and Jeff S Shamma. Consensus filters for sensor networks and distributed sensor fusion. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 6698–6703. IEEE, 2005. 4

[65] Dan R Olsen and Michael A Goodrich. Metrics for evaluating human-robot interactions. In *Proceedings of PERMIS*, volume 2003, page 4, 2003. 2.1, 3.1

[66] Dan R Olsen Jr and Stephen Bart Wood. Fan-out: measuring human control of multiple robots. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 231–238. ACM, 2004. 2.1

[67] Amanda Prorok, M Ani Hsieh, and Vijay Kumar. Formalizing the impact of diversity on performance in a heterogeneous swarm of robots. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5364–5371. IEEE, 2016. 4, 6.2.1

[68] Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 1859–1864. IEEE, 2005. 2.4, 3.3, 4, 4.1, 4.2.1, 6.2.1

[69] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics*, 21(4):25–34, 1987. 2.1, 3, 3.1, 4, 5.1

[70] Sverker Runeson and Gunilla Frykholm. Visual perception of lifted weight. *Journal of Experimental Psychology: Human Perception and Performance*, 7(4):733, 1981. 3.4

[71] David Saldana, Amanda Prorok, Shreyas Sundaram, Mario FM Campos, and Vijay Kumar. Resilient consensus for time-varying networks of dynamic agents. In *American Control Conference (ACC), 2017*, pages 252–258. IEEE, 2017. 4

[72] Navyata Sanghvi, Sasanka Nagavalli, and Katia Sycara. Exploiting robotic swarm characteristics for adversarial subversion in coverage tasks. In *Proceedings of the 16th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2017)*, pages 511–519. International Foundation for Autonomous Agents and Multiagent Systems, 2017. 4

[73] Angela P Schoellig, Hallie Siegel, Federico Augugliaro, and Raffaello DAndrea. So you think you can dance? rhythmic flight performances with quadrocopters. In *Controls and Art*, pages 73–105. Springer, 2014. 6.5

[74] William M Spears and Diana F Spears. *Physicomimetics: Physics-based swarm intelligence*. Springer Science & Business Media, 2012. 2.1, 3.1

[75] Aaron Steinfeld, Terrence Fong, David Kaber, Michael Lewis, Jean Scholtz, Alan Schultz, and Michael Goodrich. Common metrics for human-robot interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 33–40. ACM, 2006. 2.1, 3.1

[76] Frank Stürzel and Lothar Spillmann. Perceptual limits of common fate. *Vision research*, 44 (13):1565–1573, 2004. 3.4

[77] Matthew Turpin, Nathan Michael, and Vijay Kumar. Decentralized formation control with variable shapes for aerial robots. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 23–30. IEEE, 2012. 2.1, 3.1

[78] William R Uttal, Lothar Spillmann, Frank Stürzel, and Allison B Sekuler. Motion and shape in common fate. *Vision Research*, 40(3):301–310, 2000. 3.4

[79] Phillip Walker, Steven Nunnally, Mike Lewis, Andreas Kolling, Nilanjan Chakraborty, and Katia Sycara. Neglect benevolence in human control of swarms in the presence of latency. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3009–3014. IEEE, 2012. 2, 2.1, 2.6, 3, 3.2, 5

[80] Li Wang, Aaron D Ames, and Magnus Egerstedt. Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 2659–2664. IEEE, 2016. 5, 6

[81] Yorai Wardi, Magnus Egerstedt, and M Hale. Switched-mode systems: gradient-descent algorithms with armijo step sizes. *Discrete Event Dynamic Systems*, 25(4):571–599, 2015. 6.1

[82] Scott NJ Watamaniuk. Ideal observer for discrimination of the global direction of dynamic random-dot stimuli. *JOSA A*, 10(1):16–28, 1993. 3.4

[83] Scott NJ Watamaniuk, Suzanne P McKee, and Norberto M Grzywacz. Detecting a trajectory embedded in random-direction motion noise. *Vision research*, 35(1):65–77, 1995. 3.4

[84] Feng Xiao and Long Wang. State consensus for multi-agent systems with switching topologies and time-varying delays. *International Journal of Control*, 79(10):1277–1284, 2006. 4

[85] Ying Xu, Tinglong Dai, Katia Sycara, and Michael Lewis. Service level differentiation in multi-robots control. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2224–2230. IEEE, 2010. 2.1