Discrete Event Dyn Syst manuscript No.
(will be inserted by the editor)

Graph Process Specifications for Hybrid Networked Systems

Philip Twu · Patrick Martin · Magnus Egerstedt

Received: date / Accepted: date

Abstract Many large-scale multi-agent missions consist of a sequence of subtasks, each of which can be accomplished separately by having agents execute appropriate decentralized controllers. However, many decentralized controllers have network topological prerequisites that must be satisfied in order to achieve the desired effect on a system. Therefore, one cannot always hope to accomplish the original mission by having agents naively switch through executing the controllers for each subtask. This paper extends the Graph Process Specification (GPS) framework, which was presented in previous work as a way to script decentralized control sequences for agents, while ensuring that network topological requirements are satisfied when each controller in the sequence is executed. Atoms, the fundamental building blocks in GPS, each explicitly state a network topological transition. Moreover, they specify the means to make that transition occur by providing a multi-agent controller, as well as a way to locally detect the transition. Control design using GPS therefore reduces to selecting a sequence of atoms from a library to satisfy network topological requirements, and specifying interrupt conditions for switching. As an example of how to construct an atom library, the optimal decentralization algorithm is used to generate atoms for agents to track desired multi-agent motions with when the network topology is static. The paper concludes with a simulation of agents performing a drumline-inspired dance using decentralized controllers generated by optimal decentralization and scripted using GPS.

Keywords Decentralized control · formal specification · graph theoretic models · hybrid systems · network topologies

P. Twu · M. Egerstedt
Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332
E-mail: ptwu@gatech.edu, magnus@gatech.edu

P. Martin
York College of Pennsylvania, York, PA 17403 USA,
E-mail: pjmartin@ycp.edu

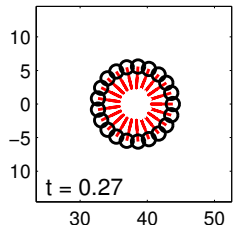
1 Introduction

As research in multi-agent systems progresses in the upcoming decades, the missions which agents are expected to undertake will also become increasingly complex. One way to reduce the complexity associated with controller design is to break the mission into a sequence of subtasks, and design decentralized controllers for completing each of the subtasks separately. To perform the entire mission, agents must then consecutively execute the controllers for each subtask in the sequence. For example, consider the task of designing controllers for a group of agents to perform the drumline-inspired multi-agent dance shown in Figure 1. One way to go about it would be to have agents switch consecutively between executing controllers designed for performing specific maneuvers, such as going into a circle or forming a GT (Georgia Tech) logo.

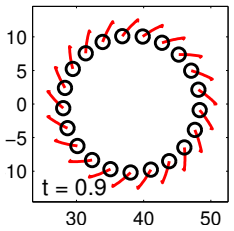
However, many decentralized controllers have network topological prerequisites that must be satisfied in order to achieve the desired effect on a system. For example, the convergence properties associated with nearest-neighbor averaging (e.g., Olfati-Saber et al (2007)) are based on the assumption that the network topology is a connected graph. Moreover, the network topology itself may be state-dependent or can change with time. Therefore, one cannot always hope to accomplish the original mission by having agents naively switch through executing the controllers for each subtask. This is because the network topology resulting from the termination of one controller may not satisfy what the next controller in the sequence needs in order to achieve its desired effects on the system when executing.

This paper presents two main theoretical contributions. The first is an extension of the Graph Process Specification (GPS) framework presented in Twu et al (2010) along with a series of detailed examples demonstrating its usage. The GPS framework acts as a way to script sequences of decentralized controllers for agents, while simultaneously ensuring that network topological requirements are satisfied for each controller in the sequence during execution. To do so, GPS builds decentralized control sequences out of fundamental building blocks called atoms. Each atom explicitly states a network topological transition (a so-called graph process as discussed in Mesbahi and Egerstedt (2010)). Moreover, every atom specifies the means to make this transition occur by providing a multi-agent controller, as well as a locally-checkable condition which allows for an agent to verify that the transition has taken place before terminating the controller.

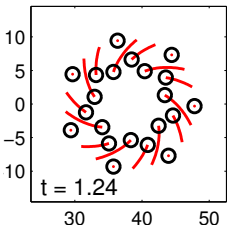
Control design in GPS reduces to selecting a sequence of atoms from a library such that each atom terminates with a network topology which the next atom in the sequence needs when starting execution in order to have the desired effect on the system. Moreover, interrupt conditions can be scripted which specify how agents will switch through the controller sequence. Together, the atom sequence and interrupts form a decentralized control strategy for agents. Upon checking that the strategy respects the network topological requirements of each controller during execution, it is downloaded onto the agents. Therefore, it is possible for agents to have special a priori designations in GPS.



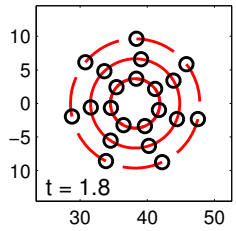
(a) Circle expands outwards.



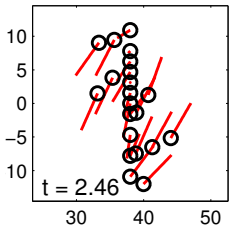
(b) Circle spirals inwards.



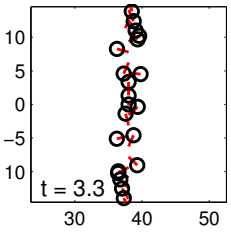
(c) Concentric circles formed.



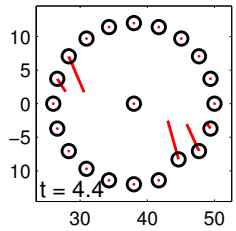
(d) Circles rotate in different directions.



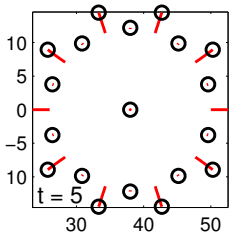
(e) Agents form vertical line.



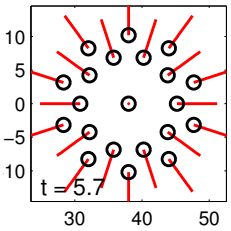
(f) Agents leave line formation.



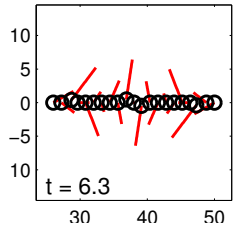
(g) Agents form a circle with center.



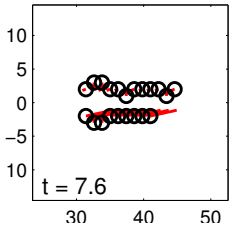
(h) Certain agents expand outwards.



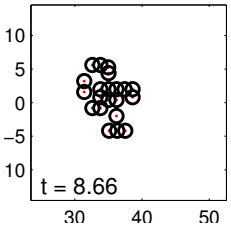
(i) Agents move towards center.



(j) Agents form a horizontal line.



(k) Agents split into two groups.



(l) Agents form a GT (Georgia Tech) logo.

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

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

The second main contribution of this paper is an example illustrating how an atom library can be constructed for use in GPS control design. In particular, the optimal decentralization algorithm from Twu and Egerstedt (2010) is presented as a way to generate atoms for agents to track desired multi-agent motions when the network topology is static. The paper concludes with a simulation showcasing agents performing a drumline-inspired dance by consecutively executing a sequence of decentralized controllers generated from optimal decentralization and scripted using GPS.

The remainder of this paper will be organized as follows: Section 2 provides background information related to GPS and the optimal decentralization algorithm. Section 3 presents the GPS framework, while Section 4 provides a detailed example of its usage in multi-agent control design. Finally, Section 5 presents the optimal decentralization algorithm as a way to generate atoms, and showcases it through a simulation.

2 Background

The idea behind the GPS framework, which was originally presented in Twu et al (2010), is related to recent developments in abstraction-based approaches to controlling groups of agents. Advances in embedded graph grammars (EGG) (e.g., McNew and Klavins (2006); Smith et al (2009)) have developed rules for a collection of agents to choose their local controllers. When a rule fires, the agents switch to the appropriate controller specified in the EGG language. Furthermore, the work in Kloetzer and Belta (2007) used linear temporal logics to specify control for a team of agents. Using this approach, the multi-agent "program" is checked for correctness before individual control laws are issued to the systems. Since a GPS specifies a sequence of decentralized controllers to be executed, it is more closely related to motion description languages (MDL) Brockett (1988); Manikonda et al (1998); Martin et al (2008). In particular, Martin et al (2008) created the MDLn framework to allow for multi-agent motion programs with networked information requirements built into the language. GPS differs from this work by considering a group-level view of the agents, rather than the execution of the individual agents' MDLn strings.

A multi-agent system using a control strategy designed through GPS acts as a hybrid system, where its low-level continuous dynamics are coupled with high-level discrete mode-switches. A sample of the rich literature available on the control of hybrid systems includes Morse (1997); Branicky et al (1998); Hedlund and Rantzer (1999); Bemporad et al (2000); Koutsoukos et al (2000);

Antsaklis (2000); Attia et al (2005); Shaikh and Caines (2007). In particular, designing a control sequence for agents by scripting a sequence of atoms in GPS is closely related to the mode sequencing problem for hybrid systems (e.g., Alamir and Attia (2004); Attia et al (2005)), except with constraints limiting which atom pairs can be executed consecutively. Moreover, GPS also allows for interrupt conditions to be specified that determine when agents switch from executing one mode to another in the sequence. Such a scenario is reminiscent of the mode scheduling problem (e.g., Axelsson et al (2008)), where optimization occurs over both the mode sequence and switching times between the modes (e.g., Shaikh and Caines (2002); Xu and Antsaklis (2002a,b); Shaikh and Caines (2003)).

The extensiveness of the atom library in GPS determines the richness of the multi-agent control sequences that can be scripted using GPS. However, one challenge in constructing such a library lies in the difficulty associated with creating decentralized controllers that have specific desired effects on a multi-agent system executing it. In general, the issue of designing decentralized controllers has received significant attention during the last decade, with two distinctly different approaches emerging. The first approach can be thought of as a bottom-up approach, where global properties are derived from local controllers. Examples of this include rendezvous and consensus controllers (e.g., Jadbabaie et al (2003); Olfati-Saber and Murray (2004); Ren and Beard (2005); Ji and Egerstedt (2007); Tanner et al (2007)), formation control (e.g., Tanner et al (2004); Eren et al (2005)), and swarming inspired controllers (e.g., Couzin and Franks (2003); Liu et al (2003); Lin et al (2004); Olfati-Saber et al (2007)). The top-down approach involves specifying a global performance metric, and then investigating when the resulting optimal controller is in fact decentralized. Examples of this view include Bamieh et al (2002); Rantzer (2007); Rotkowitz and Lall (2006); Xiao et al (2006); Motee and Jadbabaie (2008). The optimal decentralization algorithm used in this paper as an example of how one can generate atoms was originally presented in Twu and Egerstedt (2010) as a bridge between the two existing approaches to decentralized controller design. By coupling global performance metrics with parameterized constraints describing what it means for a controller to be decentralized for a system, the algorithm transforms the decentralization process into an optimization problem over the parameters.

Having placed the GPS framework into context with existing work in abstraction-based multi-agent control design, hybrid systems, and decentralized control, the next section will present technical details for the framework.

3 Graph Process Specification Formulation

3.1 Networked System Representation

Consider a collection of N agents, where the state x^i of the ith agent belongs in the differentiable manifold X, for $i \in \mathcal{N} = \{1, \ldots, N\}$. Additionally, let

$x \in X^N$ be the concatenated states of all N agents in the system, such that $x = [(x^1)^T \dots (x^N)^T]^T$. The network topology which describes the flow of information amongst agents at each instant will be represented by a vector-weighted directed graph $G = (\mathcal{N}, E, w)$, where $E \subseteq \mathcal{E} = \{(i,j) \,|\, i,j \in \mathcal{N} \text{ and } i \neq j\}$ and $(i,j) \in E$ represents a flow of information from agent i to j. Furthermore, let $w : E \to \mathbb{R}^p$, for some variable $p \in \mathbb{N}$, be a function which characterizes the information flow from one agent to another by assigning a vector to each edge. For example, in a team of mobile robots, the existence of an edge $(i,j) \in E$ means that agent j can sense agent i, while the vector-weight $w((i,j))$ may be used to describe the associated relative displacement vector which agent j senses. We will refer to this vector-weighted directed graph as the current *information flow graph* of the network. It should be noted that the vector-weight does not necessarily have to represent the information which is sensed. For example, the vector-weight can also describe the type or strength of a flow of information between two agents in the network.

We will represent the set of all possible information flow graphs that can describe the network as $\mathcal{G} = \{(\mathcal{N}, E, w) \,|\, E \subseteq \mathcal{E} \text{ and } \exists p \in \mathbb{N} \text{ where } w : E \to \mathbb{R}^p\}$. Let the mapping $s : X^N \to \mathcal{G}$ be a *graph inducing function* that takes in the states of all agents, and returns the information flow graph describing the network. Furthermore, let $\mathcal{S} = \{s \,|\, s : X^N \to \mathcal{G}\}$ be the set of all possible graph inducing functions. Note that this formulation is similar to that of connectivity graphs in Muhammad and Egerstedt (2005).

3.2 Atoms and Consistency

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

To make this information readily available, we present the concept of *atoms*, which act as the fundamental building blocks in the GPS framework. An atom contains three key pieces of information. First, each atom describes the types of multi-agent systems that it is making a statement about, i.e., those with agent dynamics in the set $\mathbb{F}$ and information flow graph given by a graph inducing function from the set $\mathbb{S}$. Second, each atom explicitly states a transition in the system's information flow graph from the initial set $\mathbb{G}$ to the final set $\mathbb{H}$. Finally, each atom describes the means by which to make this transition occur by stating a control law $\mathcal{U}$ that agents execute, as well as a condition $\mathcal{C}$ which

lets agents detect if the transition has occurred. Formally, an atom is defined as follows:

Definition 1 An *atom* $\mathcal{A}$ is a tuple given by

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

such that

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

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

An atom is *consistent* if the transition of the information flow graph from set $\mathbb{G}$ to set $\mathbb{H}$ is guaranteed to occur in finite time, and can always be detected by at least one agent in the network using the condition $\mathcal{C}$ in finite time as well. Such a guarantee is important when designing control sequences using atoms since agents should not stop executing a controller until the information flow graph makes the described transition. Ensuring that the transition occurs in finite time prevents a controller in the sequence from blocking others. Moreover, requiring that at at least one agent can detect the transition ensures that $\mathcal{C}$ is effective, and lets that agent broadcast a message throughout the network for synchronous controller switching.

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

Definition 2 An atom $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$ is *consistent* when $\forall\, f \in \mathbb{F}$, $\forall\, x_0 \in X^N$, and $\forall\, s \in \mathbb{S}$ such that $s(x_0) \in \mathbb{G}$, if

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

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

The above definition says the following: for an atom $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$, assume that the N agents have dynamics given by $f \in \mathbb{F}$ and information flow described using a graph inducing function $s \in \mathbb{S}$. Suppose that the information flow graph at some initial time $t = t_0$ belongs to the set of initial graphs $\mathbb{G}$,

and that the controller $\mathcal{U}$ is used by the agents. If $\mathcal{A}$ is consistent, then it is guaranteed that the system will evolve such that the information flow graph enters and stays in the set of final graphs $\mathbb{H}$ within finite time. Furthermore, membership of the current information flow graph in $\mathbb{H}$ can be locally detected, as indicated by when $\mathcal{C}^i \to 1$, for some agent $i \in \mathcal{N}$. A consistent atom requires that at least one agent in the network realize within finite time when the information flow graph has entered into and will stay in $\mathbb{H}$.

Note that the definition of a consistent atom allows for the control law $\mathcal{U}$ and termination condition $\mathcal{C}$ to be computed by an agent using information from anyone else in the network, even if no edge exists between the two agents in the information flow graph. To respect the limitations on inter-agent information flow as described by the network topology, it is necessary to restrict each agent's computations to use only locally available information in the network. Note that such an approach follows closely with the definition of a distributed algorithm from Lynch (1996). We therefore define a function that describes an agent computation as being *decentralized* if it satisfies the following conditions:

Definition 3 A function $\zeta : \mathcal{S} \times X^N \times \mathbb{R}_{\geq 0} \to B^N$ is *decentralized* if $\zeta^i(s, x, t) \neq \zeta^i(s, y, t)$ implies that either $x^i \neq y^i$, or there exists a $j \in \mathcal{N}$ where $(j, i) \in E$ and $x^j \neq y^j$. Here, B is some nonempty set and the edge set E comes from $s(x) = (\mathcal{N}, E, w)$. Moreover, the above must hold for all $i \in \mathcal{N}$, $s \in \mathcal{S}$, and $t \in \mathbb{R}_{\geq 0}$, and $x \in X^N$.

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

Definition 4 A *decentralized consistent atom* $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$ is a consistent atom where both $\mathcal{U}$ and $\mathcal{C}$ are decentralized.

Before proceeding onwards, we will solidify the concepts presented thus far through an example of constructing a decentralized consistent atom for agents to perform nearest-neighbor averaging. It should be noted that a rich literature exists on nearest-neighbor averaging controllers for multi-agent systems, each with their own associated merits and demerits. The controller used in this example was chosen merely to illustrate the process of encapsulating a decentralized controller into a decentralized consistent atom.

Example 1 Suppose that in a team of N agents, the ith agent has state $x^i \in X = \mathbb{R}^2$ that describes its planar position in Cartesian coordinates, for all $i \in \mathcal{N}$. Since we are interested in focusing on high-level coordination strategies,

agents will be treated as point particles with single integrator dynamics $f_{\mathcal{I}} : \mathbb{R}^{2N} \times \mathbb{R}^{2N} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{2N}$, where

$$\dot{x} = f_{\mathcal{I}}(x, u, t) = u. \tag{1}$$

Furthermore, let

$$\mathbb{F}_{\mathcal{I}} = \{f_{\mathcal{I}}\} \tag{2}$$

be the set containing the agents' single integrator dynamics.

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

$$\mathbb{S}_{\Delta}(\delta) = \{s_{\Delta}(\delta)\} \tag{3}$$

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

The goal of nearest-neighbor averaging is to have agents within the network reach consensus with one another by converging to the same state. In our case, because an agent's state is its position, nearest-neighbor averaging will drive the agents to meet at the same location. To do so, each agent calculates its control using only sensed relative displacement vectors to its neighbors. For the chosen graph inducing function and agent dynamics, Ji and Egerstedt (2007) present the following decentralized controller for performing nearest-neighbor averaging while maintaining network connectivity:

$$\mathcal{U}^i_{avg}(\delta)(s_{\Delta}(\delta), x, t) = -\sum_{j \in N_{\rho}(i)} \frac{2\delta - ||x^i - x^j||}{(\delta - ||x^i - x^j||)^2}(x^i - x^j), \tag{4}$$

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

Ji and Egerstedt (2007) state that in a network where every pair of nodes has a path that connects them (i.e., a connected graph) consisting of edges with weights less than δ, then the nearest-neighbor averaging controller will drive all agent states to the same value asymptotically. Consequently, the

network topology will become and stay as K_N, the complete graph with N nodes (i.e., an edge exists between every pair of agents) in finite time. Thus, the set of initial information flow graphs for the nearest-neighbor averaging atom is given by $\mathbb{G}_{avg}$, where

$$\mathbb{G}_{avg}(\delta) = \{(\mathcal{N}, E, w) \mid \exists e \subseteq E \text{ where } (\mathcal{N}, e) \text{ is connected and } w((i,j)) < \delta, \forall (i,j) \in e\}. \tag{5}$$

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

$$\mathcal{C}^i_{avg}(\lambda)(s,x,t) = \begin{cases} 1 \text{ , if } N(i) = \mathcal{N} - \{i\} \text{ and } ||x^i - x^j|| \leq \frac{\lambda}{2} \, \forall \, j \in N(i) \\ 0 \text{ , otherwise,} \end{cases} \tag{6}$$

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

$$\mathbb{H}_{avg}(\lambda) = \{(\mathcal{N}, E, w) \mid (\mathcal{N}, E) = K_N \text{ and } w((i,j)) \leq \lambda \, \forall (i,j) \in E\}. \tag{7}$$

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

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

Lemma 1 *The atom for agents to perform nearest-neighbor averaging:*

$$\mathcal{A}_{avg}(\lambda, \delta) = (\mathbb{S}_{\Delta}(\delta), \mathbb{F}_{\mathcal{I}}, \mathbb{G}_{avg}(\delta), \mathbb{H}_{avg}(\lambda), \mathcal{U}_{avg}(\delta), \mathcal{C}_{avg}(\lambda)), \tag{8}$$

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

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

3.3 Modes and Composability

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

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

Definition 5 A *mode* is denoted by the tuple

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

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

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

It is important to note that there is nothing which says that the interrupt condition ξ cannot be blocking, e.g., if the conditions for setting off the interrupt contradict those required for atom consistency. This design choice was made to give the user the most flexibility when writing scripts for agents. Such a choice follows the approach adopted by many mainstream computer

programming languages (e.g., the ability to write infinite-loops and deadlock scenarios in Java), as well as abstraction-based motion-programming languages (e.g., blocking interrupt conditions in MDL and MDLe). Therefore, it is at the discretion of the user to avoid such blocking scenarios whenever specifying interrupts.

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

Definition 6 The mode $\mathcal{M}_1 = (\mathcal{A}_1, \xi_1)$ is *composable* with the mode $\mathcal{M}_2 = (\mathcal{A}_2, \xi_2)$, where $\mathcal{A}_1 = (\mathbb{S}_1, \mathbb{F}_1, \mathbb{G}_1, \mathbb{H}_1, \mathcal{U}_1, \mathcal{C}_1)$ and $\mathcal{A}_2 = (\mathbb{S}_2, \mathbb{F}_2, \mathbb{G}_2, \mathbb{H}_2, \mathcal{U}_2, \mathcal{C}_2)$, if $\mathbb{H}_1 \subseteq \mathbb{G}_2$. We will denote this property by $\mathcal{M}_1 \prec \mathcal{M}_2$.

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

3.4 Graph Process Specifications

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

Definition 7 A *Graph Process Specification (GPS)* is a tuple given by

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

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

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

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

Definition 8 A GPS $((x_0, s^*, f^*), (\mathcal{M}_1, \ldots, \mathcal{M}_m))$ is *executable* if

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

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

3.5 Graph Process Specification Executions

To illustrate how agents will behave when following a control sequence scripted by an executable GPS, we will formally describe its execution. We start by defining a variant of the *hybrid time sets* used in Johansson et al (1999) to describe the time intervals in which each mode of the GPS is being executed:

Definition 9 A *hybrid time set* is a sequence of intervals $Q = \{q_1, \ldots, q_w\}$, for some $w \in \mathbb{N}$, such that

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

Hybrid time sets are used to describe the execution of a GPS similar to how Johansson et al (1999) uses them to describe the execution of a hybrid system: as a set of requirements on the state trajectory. Therefore, a state trajectory

is either accepted or rejected as an execution of the GPS. To be an execution of a GPS, the state trajectory must begin at the initial condition specified in the GPS. The state evolution in each mode must be driven by the controller in that mode's consistent atom, as applied to the agent dynamics. Lastly, each mode terminates as soon as any agent detects that the information flow graph has entered into the set of final graphs and that the interrupt conditions are satisfied as well. Although the end of a mode is detected by a single agent, all agents switch modes simultaneously.

Definition 10 Given an executable GPS, $((x_0, s^*, f^*), (\mathcal{M}_1, \ldots, \mathcal{M}_m))$, its *execution* is a pair (Q, x), where $Q = \{q_1, \ldots, q_{\tilde{m}}\}$ is a hybrid time set with $\tilde{m} \leq m$ and $z_1 = 0$. If $\tilde{m} < m$, then $z'_{\tilde{m}} = \infty$, while if $\tilde{m} = m$, then we allow for $z'_{\tilde{m}} \leq \infty$. Additionally, $x(t)$ is a state trajectory defined on either $t \in [0, \infty)$ if $z'_{\tilde{m}} = \infty$, or on $t \in [0, z'_{\tilde{m}}]$ if $z'_{\tilde{m}} < \infty$, such that

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

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

   If $z'_{\tilde{m}} < \infty$, then the above also holds for $k = \tilde{m}$.
4. For each $k = 1, \ldots, \tilde{m}$, $\nexists\, t \in q_k - \{z'_k\}$ such that

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

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

This definition describes an execution of a GPS in the following way: the state trajectory $x(t)$ of the agents starts at the initial condition x_0 at time $t = 0$. Given that the GPS contains a sequence of m modes, q_k corresponds to the time that mode k is being executed, for $k = 1, \ldots, \tilde{m}$, where $\tilde{m} \leq m$. In the kth mode, as indicated by when $t \in q_k$, the agent state dynamics f^* uses the controller $\mathcal{U}_k$, as supplied by $\mathcal{A}_k$. The kth mode stops and switches to the $k + 1$th mode in the sequence (or stops the execution of the GPS if $k = m$) the instant $t = z'_k$. This corresponds to the first time $t \in q_k$ when both $\mathcal{C}_k^i \to 1$ and $\xi_k^i \to 1$, for any agent $i \in \mathcal{N}$. Finally, since the end of the kth mode depends on a user defined interrupt mapping ξ_k, it is possible that the interrupt never fires, causing the kth mode to continue executing forever. Such a blocking scenario is why we allow for $\tilde{m} \leq m$. Figure 2 provides an illustration showing how the execution of an executable GPS with three modes is viewed as a hybrid system.

In this section, we have defined the tools which the GPS framework is composed of, as well as the execution of an executable GPS. In the next section, we will give a detailed example of how one can go about writing a script for a decentralized control sequence in GPS that makes agents switch between multiple formations.

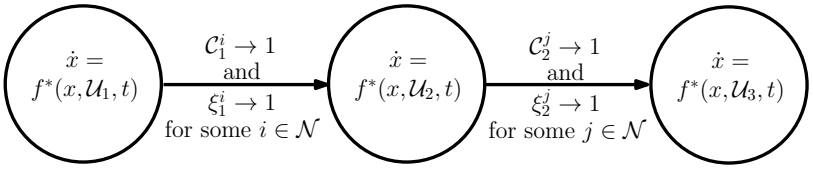


Fig. 2 An illustration showing how the execution of an executable GPS with three modes can be viewed as a hybrid system.

4 Graph Process Specification Examples

In this section, we provide an example of how GPS can be used to script a decentralized control sequence for a multi-agent system that makes agents first go into a line formation, and then switch into a circle formation. The process of encapsulating the relevant multi-agent controllers into decentralized consistent atoms will be illustrated in detail. Two scripts written as GPS's will be created using those atoms: the original which is not executable, and a revised one which is made executable through a mode insertion. Note that this section builds off of the nearest-neighbor averaging example in Example 1. Therefore, all assumptions and definitions made previously will still hold true in this section.

4.1 Connectedness-Preserving Formation Control Laws

We will use the connectedness-preserving formation control law from Ji and Egerstedt (2007) throughout this example as a way to have agents move into desired formations. It should be noted that many controllers exist in literature for formation control in multi-agent systems, each with their own respective merits and demerits. We use the formation control law from Ji and Egerstedt (2007) here simply as an example to illustrate how one can encapsulate an existing decentralized multi-agent controller into a decentralized consistent atom.

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

Have $\tau = [(\tau^1)^T \ \dots \ (\tau^N)^T]^T$ be the vector of concatenated target points. The shorthand notation $d_{ij} = \tau^i - \tau^j$, for all $i, j \in \mathcal{N}$, will be used to represent the displacement vector between any two agents i and j in the desired

formation. Furthermore, let $l_{ij}(t) = x^i(t) - x^j(t)$, for all $i, j \in \mathcal{N}$, be the actual displacement vector between agents i and j at time t. The controller which allows for agents to achieve formations while maintaining network connectivity will now be presented.

Theorem 1 *Suppose a multi-agent system has graph inducing function $s_\Delta(\delta)$ and dynamics $f_\mathcal{I}$. Let the information flow graph induced by the target points be given by $s_\Delta(\delta)(\tau) = (\mathcal{N}, E_d, w_d)$, where $(\mathcal{N}, E_d)$ is connected and $w_d((i,j)) < \delta$ for all $(i,j) \in E_d$. Furthermore, have the graph induced by the agent states at time t be $s_\Delta(\delta)(x(t)) = (\mathcal{N}, E(x(t)), w(x(t)))$. Ji and Egerstedt (2007) state that if at some initial time $t = t_0$:*

1. *$E_d \subseteq E(x(t_0))$,*
2. *$||l_{ij}(t_0)|| \leq \epsilon^*$ for some specific $\epsilon^* > 0$ (see Ji and Egerstedt (2007) for details), for all $(i,j) \in E_d$,*

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

$$\begin{aligned} &\mathcal{U}^i_{form}\left(E_d, \tau, \delta\right)\left(s_\Delta(\delta), x, t\right) \\ &= -\sum_{j \in N_{G_d}(i)} \frac{2\left(\delta - ||d_{ij}||\right) - ||l_{ij}(t) - d_{ij}||}{\left(\delta - ||d_{ij}|| - ||l_{ij}(t) - d_{ij}||\right)^2}\left(d_{ij} - l_{ij}(t)\right) \end{aligned} \tag{9}$$

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

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

Next, we will specialize this controller to make agents go into line and circle formations, as well as encapsulate them within decentralized consistent atoms to be used in the GPS framework.

4.2 Line-Formation Decentralized Consistent Atom

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

$$\tau^1_{line} = [0 \;\; 0]^T \text{ and } \tau^{i+1}_{line} = \tau^i_{line} - [0.9\delta \;\; 0]^T, \tag{10}$$

for $i = 1, \ldots, N-1$. The information flow graph resulting from applying $s_{\Delta}(\delta)$ to τ_{line} is the line graph $G_{line} = (\mathcal{N}, E_{line}, w_{line})$ where $E_{line} = \{(i, i+1) \,|\, i = 1, \ldots, N-1\}$ and $w_{line}((i,j)) = 0.9\delta$, for all $(i,j) \in E_{line}$. To better visualize the desired line formation, the location of the target points, along with the corresponding network topology, are illustrated in Figure 3 for $N = 6$ and $\delta = 1$.

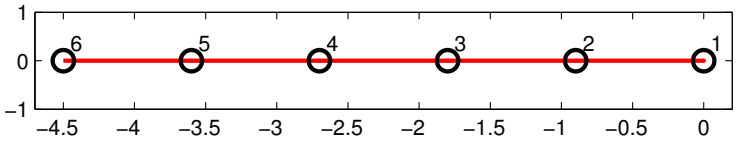


Fig. 3 The target points τ_{line} and the network topology $(\mathcal{N}, E_{line})$ for the line formation, with $N = 6$ and $\delta = 1$.

To have agents successfully use the formation control law in Theorem 1, we require that the information flow graph induced by the agent states initially belong to

$$\begin{aligned}\mathbb{G}_{line}(\epsilon^*_{line}) = \{(\mathcal{N}, E, w) \,|\, & E_{line} \subseteq E \text{ and} \\ & w((i,j)) \leq \epsilon^*_{line} \text{ for all } (i,j) \in E_{line}\}\,, \end{aligned} \tag{11}$$

where ϵ^*_{line} is chosen appropriately. The desired line formation can then be achieved by specializing the formation control law (9) to create a new control law $\mathcal{U}_{line}(\delta)$, where

$$\mathcal{U}^i_{line}(\delta)(s,x,t) = \mathcal{U}^i_{form}\left(E_{line}, \tau_{line}, \delta\right)(s,x,t), \tag{12}$$

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

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

$$\begin{aligned}\mathbb{H}_{line}(\delta) = \{(\mathcal{N}, E, w) \,|\, & E_{line} \subseteq E, w((i,j)) < \delta \text{ for all } (i,j) \in E_{line}, \\ & \text{and } (1,j) \notin E, \forall\, j \neq 2\}\,, \end{aligned} \tag{13}$$

In the set of final graphs above, the requirements on the edge weights are guaranteed to be met by the connectedness-preserving nature of the controller. For simplicity, we will let agent 1 be the only one that can detect the transition

of the information flow graph into the set $\mathbb{H}_{line}(\delta)$. To do so, agent 1 will be using the decentralized function $\mathcal{C}_{line}$, where

$$\mathcal{C}^1_{line}(s,x,t) = \begin{cases} 1 \text{ , if } N(1) = \{2\} \\ 0 \text{ , otherwise,} \end{cases} \tag{14}$$

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

$$\mathcal{C}^i_{line}(s,x,t) = 0, \text{ for } i = 2,\ldots,N. \tag{15}$$

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

Lemma 2 *The atom for driving agents into a line formation:*

$$\mathcal{A}_{line}(\epsilon^*_{line},\delta) = (\mathbb{S}_\Delta(\delta), \mathbb{F}_\mathcal{I}, \mathbb{G}_{line}(\epsilon^*_{line}), \mathbb{H}_{line}(\delta), \mathcal{U}_{line}(\delta), \mathcal{C}_{line}), \tag{16}$$

where ϵ^*_{line} is chosen to satisfy Theorem 1, is a decentralized consistent atom.

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

4.3 Circle-Formation Decentralized Consistent Atom

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

$$\tau^1_{circ} = [0 \;\; 0]^T \tag{17}$$

and

$$\tau^{i+1}_{circ} = \tau^i_{circ} + Rot\left(\frac{2\pi}{N}\right)^i [0.9\delta \;\; 0]^T, \tag{18}$$

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

$$Rot(\theta) = \begin{bmatrix} cos\,(\theta) & -sin\,(\theta) \\ sin\,(\theta) & cos\,(\theta) \end{bmatrix}. \tag{19}$$

Furthermore, have $\tau_{circ} = [(\tau^1_{circ})^T \;\ldots\; (\tau^N_{circ})^T]^T$ be the concatenated target points.

The information flow graph resulting from applying $s_\Delta(\delta)$ to τ_{circ} is the cycle graph $G_{circ} = (\mathcal{N}, E_{circ}, w_{circ})$ where $E_{circ} = \{(N,1)\} \cup \{(i,i+1) \,|\, i =$

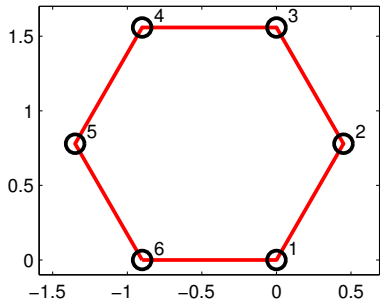


Fig. 4 The target points τ_{circle} and the network topology $(\mathcal{N}, E_{circle})$ for the circle formation, with $N = 6$ and $\delta = 1$.

$1, \ldots, N-1\}$ and $w_{circ}((i,j)) = 0.9\delta$, for all $(i,j) \in E_{circ}$. To better visualize the desired circle formation, Figure 4 shows the locations of the target points in τ_{circ}, along with the corresponding network topology, for $N = 6$ and $\delta = 1$.

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

$$\mathbb{G}_{circ}(\epsilon^*_{circ}) = \{(\mathcal{N}, E, w) \,|\, E_{circ} \subseteq E \text{ and } w((i,j)) \leq \epsilon^*_{circ} \text{ for all } (i,j) \in E_{circ}\}. \tag{20}$$

We specialize the control law (9) to achieve the desired circle formation by defining the controller $\mathcal{U}_{circ}(\delta)$, where

$$\mathcal{U}^i_{circ}(\delta)(s,x,t) = \mathcal{U}^i_{form}\left(E_{circ}, \tau_{circ}, \delta\right)(s,x,t). \tag{21}$$

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

$$\mathbb{H}_{circ}(\delta) = \{(\mathcal{N}, E, w) \,|\, E_{circ} \subseteq E, w((i,j)) < \delta \text{ for all } (i,j) \in E_{circ}, \text{ and } N(1) = \{2,6\}\}, \tag{22}$$

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

$$\mathcal{C}^1_{circ}(s,x,t) = \begin{cases} 1 \text{ , if } N(1) = \{2,6\} \\ 0 \text{ , otherwise,} \end{cases} \tag{23}$$

and

$$\mathcal{C}^i_{circ}(s,x,t) = 0, \text{ for } i = 2, \ldots, N. \tag{24}$$

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

Lemma 3 *The atom for driving agents to a circle formation:*

$$\mathcal{A}_{circ}(\epsilon^*_{circ}, \delta) = (\mathbb{S}_{\Delta}(\delta), \mathbb{F}_{\mathcal{I}}, \mathbb{G}_{circ}(\epsilon^*_{circ}), \mathbb{H}_{circ}(\delta), \mathcal{U}_{circ}(\delta), \mathcal{C}_{circ}), \tag{25}$$

where ϵ^*_{circ} is chosen to satisfy Theorem 1, is a decentralized consistent atom.

Proof The proof is identical to that for $\mathcal{A}_{line}(\epsilon^*_{line}, \delta)$ in Lemma 2.

4.4 Locally Executable GPS Example

Thus far in the paper, a total of three decentralized consistent atoms have been constructed: nearest-neighbor averaging (Lemma 1), line formation (Lemma 2), and circle formation (Lemma 3). Together, these three atoms make a simple atom library which we can use in this example for scripting decentralized control sequences for agents. Consider now, the task of designing a control sequence which will make agents first go into a line formation, and then switch into a circle formation. We will show how to use the atoms already existing in our simple atom library to design a decentralized control strategy to achieve this.

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

$$x_0^1 = [0 \ \ 0]^T, \quad x_0^2 = [0.25 \ \ 0.25]^T, \quad x_0^3 = [0.25 \ \ 0.55]^T,$$
$$x_0^4 = [0 \ \ 0.37]^T, \ x_0^5 = [-0.37 \ \ 0.37]^T, \ x_0^6 = [-0.25 \ \ 0.75]^T.$$

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

Now that we know the system which we will be designing a control sequence for, the next step is to specify a sequence of atoms for the agents. Note that the overall mission for the agents can be broken down into two subtasks: form a line, and then form a circle. Our first attempt at constructing an atom sequence is to use the naive approach of using one atom to accomplish each of the subtasks, and simply executing the two back to back. Letting ϵ^*_{line} and ϵ^*_{circ} from Lemmas 2 and 3 equal 0.8, we get that $s_{\Delta}(1)(x_0) \in \mathbb{G}_{line}(0.8)$ and $s_{\Delta}(1)(x_0) \in \mathbb{G}_{circ}(0.8)$. The first proposed atom sequence is therefore given by $\mathcal{A}_{line}(0.8, 1)$, followed immediately by $\mathcal{A}_{circ}(0.8, 1)$.

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

$$\mathcal{M}_{line} = (\mathcal{A}_{line}(0.8, 1), \xi_{line}) \text{ and } \mathcal{M}_{circ} = (\mathcal{A}_{circ}(0.8, 1), \xi_{circ}). \tag{26}$$

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

$$GPS_1 = ((x_0, s_\Delta(1), f_\mathcal{I}), (\mathcal{M}_{line}, \mathcal{M}_{circ})) \tag{27}$$

Checking GPS_1 reveals that it is *not executable* because the mode $\mathcal{M}_{line}$ does not compose with $\mathcal{M}_{circ}$. This is because any graph in $\mathbb{H}_{line}(1)$ has agent 2 being the only neighbor of agent 1, whereas all graphs in $\mathbb{G}_{circ}(0.8)$ require that agents 1 and 6 be neighbors as well. Furthermore, some graphs belonging to $\mathbb{H}_{line}(1)$ have edge weights which are too large to belong in $\mathbb{G}_{circ}(0.8)$.

To fix these problems, we will consider inserting a mode between $\mathcal{M}_{line}$ and $\mathcal{M}_{circ}$ that adds additional edges to the information flow graph and decrease all the edge weights. Fortunately, the decentralized consistent atom for nearest-neighbor averaging from Example 1 does exactly what is needed in this situation. Using $\mathcal{A}_{avg}(\lambda, \delta)$, we can define the decentralized mode

$$\mathcal{M}_{avg} = (\mathcal{A}_{avg}(0.8, 1), \xi_{avg}), \tag{28}$$

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

Lemma 4 *The graph process specification*

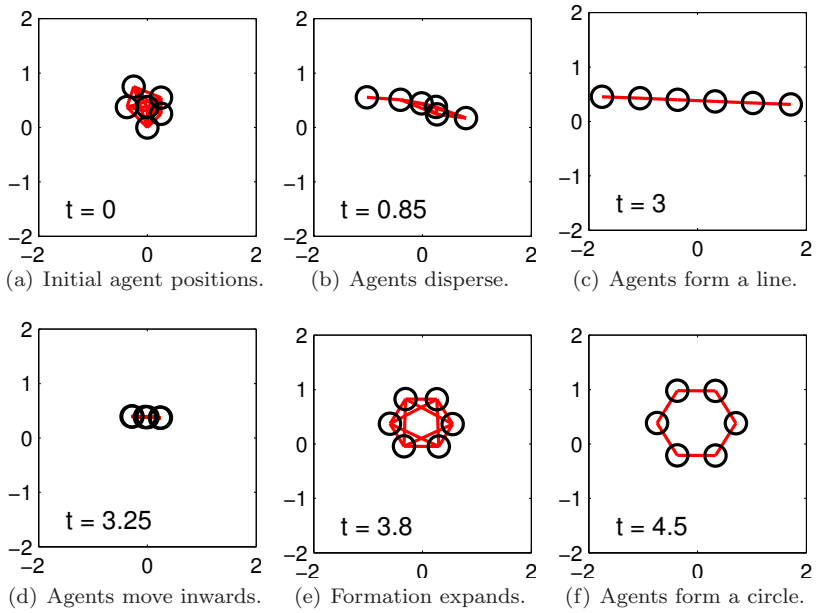
$$GPS_2 = ((x_0, s_\Delta(1), f_\mathcal{I}), (\mathcal{M}_{line}, \mathcal{M}_{avg}, \mathcal{M}_{circ})) \tag{29}$$

is locally executable.

Proof First, note that $s_\Delta(1)$ and $f_\mathcal{I}$ belong to the respective graph inducing function and agent dynamics sets in the decentralized consistent atoms of all three modes. Since $s_\Delta(1)(x_0)$ gives a complete graph where all edge weights are less than 0.8, $s_\Delta(1)(x_0) \in \mathbb{G}_{line}(0.8)$. Noticing that each graph in $\mathbb{H}_{line}(1)$ has a line graph as a subgraph, with edge weights strictly less than 1, we see that $\mathbb{H}_{line}(1) \subseteq \mathbb{G}_{avg}(1)$. Lastly, because $\mathbb{H}_{avg}(0.8)$ only contains complete graphs with edge weights less than or equal to 0.8, each graph contains the required cycle subgraph to belong to $\mathbb{G}_{circ}(0.8)$ and so $\mathbb{H}_{avg}(0.8) \subseteq \mathbb{G}_{circ}(0.8)$. These checks show that GPS_2 is executable. Furthermore, since each of the modes in GPS_2 are decentralized modes, it is also locally executable.

A simulation of agents executing the decentralized controller sequence scripted in the locally executable GPS_2 is shown in Figure 5, for $N = 6$ and $\delta = 1$.

The strategy of inserting a mode containing the atom $\mathcal{A}_{avg}(\lambda, \delta)$ into GPS_1's mode sequence to form the executable GPS_2 turns out to be a useful strategy which applies to many scenarios. Therefore, $\mathcal{A}_{avg}(\lambda, \delta)$ can be thought of as a "universal glue" for certain pairs of modes which are not composable.



(a) Initial agent positions. (b) Agents disperse. (c) Agents form a line.

(d) Agents move inwards. (e) Formation expands. (f) Agents form a circle.

Fig. 5 Simulation of agents executing the decentralized controller sequence scripted using the locally executable GPS_2, as given in (29), for $N = 6$ and $\delta = 1$. The location of the agents are marked by O's and the lines indicate edges in the induced graph.

In particular, the final set of graphs in the first mode must be a subset of $\mathbb{G}_{avg}(\delta)$. Such a requirement is very reasonable since all that is required is that the first mode terminates with an information flow graph that remains connected even in the presence of small perturbations to inter-agent displacements. Since $\mathcal{A}_{avg}(\lambda, \delta)$ adds edges to the information flow graph and decreases the edge weights, the set of initial graphs in the second mode must include all graphs after a certain number of edges have been added and all edge weights have fallen below some threshold. Such a property is described in the definition below.

Definition 11 Suppose there exists a nonempty set of information flow graphs $\mathbb{G} \subseteq \mathcal{G}$, whose members all have vector-weight functions w that return scalar values. $\mathbb{G}$ is *inclusive* if $(\mathcal{N}, E, w) \in \mathbb{G}$ implies that $(\mathcal{N}, \hat{E}, \hat{w}) \in \mathbb{G}$ as well, where $E \subseteq \hat{E}$ and $\hat{w}((i,j)) \leq w((i,j))$ for all $(i,j) \in E$.

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

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

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

To summarize the example presented in this section, we started out with a mission consisting of two subtasks: having agents first go into a line formation, and then switch to a circle formation. The formation control law from Ji and Egerstedt (2007) was used to create decentralized consistent atoms containing controllers that drive agents into each of the formations separately. Using the GPS framework, a script written using the naive approach of executing the controllers for achieving the two formations back to back was shown to not be executable. This was because the information flow graph of the system upon terminating the line formation controller did not allow for the circle formation controller to immediately start executing afterwards. However, the missing transition in the system's information flow graph could be supplied by inserting a mode for nearest-neighbor averaging between the two existing modes. The resulting GPS was checked to be locally executable and a simulation was shown of agents executing the scripted decentralized control sequence to accomplish the original mission.

5 Example of Generating Decentralized Consistent Atoms

To use GPS effectively in scripting control sequences for multi-agent systems, it is necessary to have a library of decentralized consistent atoms available to construct mode sequences from. While the previous section demonstrated how to use such an atom library, little has been said on how such a library can be constructed in the first place. As can be seen in the examples thus far, many challenges may arise when attempting to construct a decentralized consistent atom. First is the difficulty associated with decentralized controller design itself. Second is in interpreting the assumptions made when designing the controller from a network topological perspective, as well as understanding how the information flow graph is effected during the controller's execution. Finally, the third challenge lies in designing termination conditions that allow agents to detect when a transition in the network's information flow graph has occurred, while using only locally available information.

In this section, we will present an example that illustrates the concept of generating decentralized controllers from centralized specifications, thereby focusing on the first of the three challenges that were highlighted. In particular, we will present the optimal decentralization algorithm from Twu and Egerstedt (2010) as a way to generate decentralized controllers for agents to track desired multi-agent motions which have been defined on the agent trajectory level. Then, we connect those results with the GPS framework by showing how the generated controllers can be encapsulated into decentralized consistent atoms for populating an atom library with. Note that the main focus of this

section lies in how the products of decentralized control design methods can be encapsulated into decentralized consistent atoms for use in GPS. Therefore, we will restrict our attention to multi-agent systems where the network topology is static, and switching conditions for controllers are timer-based (as opposed to state-dependent).

It should be noted that a general theory on generating decentralized control laws for agents in a setting with dynamic network topologies is still an unsolved problem. Moreover, how to design algorithms which allow for agents to determine the overall network topology by using only local-available conditions (i.e., state dependency rather than timer-based conditions) is an open research question as well. Although allowing for multi-hop inter-agent communication may help get around this issue, the focus of this section is mainly on decentralized controller design. Therefore, introducing any new agent capabilities at this point will only complicate matters and draw focus away from the intended theoretical contribution of this section.

5.1 Optimal Decentralization

The optimal decentralization algorithm generates a decentralized control strategy for agents to track a desired multi-agent motion that has been defined on the agent trajectory level. In particular, the algorithm generates a sequence of decentralized controllers, as well as switching times for transitioning through the sequence. Therefore, the output of the algorithm can be used to generate both a sequence of atoms, as well as locally-checkable timing requirements that describe when each controller should terminate execution. To do so, it bridges both the top-down and bottom-up approaches to decentralized controller design that was described in Section 2. A global performance metric is coupled with parameterized constraints describing what a decentralized controller is for the system, and an optimization problem is solved to find the parameters which allows agents to track the desired motion the best.

Similar to the examples from the previous section, we assume that the multi-agent system of interest has N agents. The state $x^i \in X = \mathbb{R}^2$ describes the position of the ith agent using Cartesian coordinates, for all $i \in \mathcal{N}$, and each agent has single integrator dynamics $f_{\mathcal{I}}$. Once again, we assume that each agent can sense the relative displacement vector to each of its neighbors in the network. However, we will restrict our attention in this algorithm to systems where the inter-agent information flow is given by a static network topology. Such a setup could, for example, describe a situation in which the sensing radius of each agent is much larger than the area in which their movement is confined to. In such a scenario, each agent only chooses to keep track of a select few other agents in the network for complexity and scalability reasons. The information flow graph used in this system will be given by the graph inducing function $s_\sigma(x) = (\mathcal{N}, E_\sigma, w_\sigma(x))$, where E_σ is fixed and $w_\sigma(x)((i,j)) = x^i - x^j$ gives the relative displacement between neighboring agents, for all $(i,j) \in E_\sigma$.

Given some mission defined on the agent trajectory level which starts at $t = 0$ and ends at $t = T$, we wish to generate an executable sequence of m decentralized controllers that tracks the given trajectory. The multi-agent system will transition through the controller sequence using global clock-based switching, where the kth mode occurs during the time interval $[\tau_{k-1}, \tau_k)$, for $k = 1, \ldots, m$, and where the global switching times must satisfy the constraint

$$0 = \tau_0 \leq \tau_1 \leq \ldots \leq \tau_{m-1} \leq \tau_m = T. \tag{30}$$

In each mode, agents compute their control as a linear combination of scaled and rotated displacement vectors between itself and each of its neighbors. The dynamics for the ith agent operating in the kth mode are therefore

$$\dot{x}^i = -\sum_{j \in N(i)} r_{ijk}\, Rot\left(\theta_{ijk}\right)\left(x^i - x^j\right), \quad \forall t \in [\tau_{k-1}, \tau_k), \tag{31}$$

with $r_{ijk} \in \mathbb{R}$ and $\theta_{ijk} \in [0, 2\pi)$ parameterizing the scaling and rotation, respectively, of the displacement vector between agents i and j. Notice that (31) gives a general form for a class of decentralized controllers. By optimally selecting the mode parameters r_{ijk} and θ_{ijk} , as well as the global switching times τ_k, a decentralized controller sequence can be generated to minimize the tracking error with the given target trajectory.

When performing optimizations on the parameters r_{ijk} and θ_{ijk}, it is helpful to use the following change of variables:

$$a_{ijk} = r_{ijk} \cos\left(\theta_{ijk}\right) \tag{32}$$

$$b_{ijk} = r_{ijk} \sin\left(\theta_{ijk}\right), \tag{33}$$

and optimize over a_{ijk} and b_{ijk} instead. This helps to avoid any issues associated with the cyclic nature of θ_{ijk}. Therefore, by letting

$$M_{ijk} = \begin{bmatrix} a_{ijk} & -b_{ijk} \\ b_{ijk} & a_{ijk} \end{bmatrix}, \tag{34}$$

the agent dynamics in (31) can be rewritten as

$$\dot{x}^i = -\sum_{j \in N(i)} M_{ijk}\left(x^i - x^j\right), \quad \forall t \in [\tau_{k-1}, \tau_k), . \tag{35}$$

Moreover, to make optimizing over all agents' parameters easier, the entire multi-agent system's dynamics can be collected together in matrix form. First, the parameters a_{ijk} and b_{ijk} can be grouped together, for each mode k, into vectors

$$a_k = [\ldots, a_{ijk}, \ldots]^T \text{ and } b_k = [\ldots, b_{ijk}, \ldots]^T. \tag{36}$$

Define the $(2N \times 2N)$ adjacency matrix $A_k(a_k, b_k)$ associated with the kth mode, in terms of (2×2) blocks, by

$$A_{ijk}(a_k, b_k) = \begin{cases} M_{ijk} & \text{, if } (j,i) \in E_\sigma \\ 0 & \text{, otherwise.} \end{cases} \tag{37}$$

Next, let the $(2N \times 2N)$ degree matrix $D_k(a_k, b_k)$ associated with the kth mode also be defined in terms of (2×2) blocks by

$$D_{ijk}(a_k, b_k) = \begin{cases} \sum_{z|(z,i)\in E_\sigma} M_{izk} \text{ , if } i = j \\ 0 \qquad\qquad\qquad \text{, otherwise.} \end{cases} \tag{38}$$

Finally, define the weighted Laplacian matrix $L_k(a_k, b_k)$ associated with the kth mode as

$$L_k(a_k, b_k) = D_k(a_k, b_k) - A_k(a_k, b_k). \tag{39}$$

The evolution of x, the concatenated states of all N agents in the multi-agent system, can now be described by the dynamics

$$\dot{x} = -L_k(a_k, b_k)x, \quad \forall t \in [\tau_{k-1}, \tau_k), \tag{40}$$

for each mode $k = 1, \ldots, m$.

Assuming that the agents have initial state $x(0) = x_0$, the task of generating a sequence of m decentralized controllers to track some target trajectory $x_d : [0, T] \to \mathbb{R}^{2N}$ can be treated as an optimal control problem. In this case, the objective is to optimally choose the parameters a_k and b_k for each mode $k = 1, \ldots, m$, as well as the global switching times $\tau_1, \ldots, \tau_{m-1}$, so as to minimize the cost functional

$$J = \frac{1}{2} \int_0^T ||x(t) - x_d(t)||^2 \, dt \tag{41}$$

for a system with dynamics (40). Extra care must be taken to ensure that the global switching times satisfy the constraints in (30).

5.2 Optimization of Parameterized Modes

To optimize a_k and b_k in each mode $k = 1, \ldots, m$, so as to minimize J, we will first derive costate dynamics and optimality conditions for a general switched autonomous system with parameterized modes. Those results will then be specialized for our multi-agent system of interest in (40).

Consider a system that evolves as a switched autonomous system starting at time $t = 0$, and ending at time $t = T$, with m modes and $m - 1$ switching times. Each of the modes' dynamics are given by the function f but are parameterized by different scalar parameters γ_k, for each mode $k = 1, \ldots, m$. The switching times are $\tau_1, \ldots, \tau_{m-1}$ satisfying (30), with the kth mode occurring in the time interval $[\tau_{k-1}, \tau_k)$. Letting

$$\Gamma = [\gamma_1, \ldots, \gamma_m]^T \tag{42}$$

be the vector containing the parameters for each mode, the dynamics of the system are given by

$$\dot{x} = F(x, \Gamma, t) \tag{43}$$

with

$$F\left(x,\Gamma,t\right)=f\left(x,\gamma_k\right)\quad\forall t\in[\tau_{k-1},\tau_k). \tag{44}$$

The objective is to choose Γ so as to minimize the generalized cost functional

$$J=\int_0^T H\left(x\left(t\right)\right)\,dt. \tag{45}$$

Theorem 3 *The optimality condition for each γ_k in (44) with respect to cost (45) is*

$$\frac{\partial J}{\partial\gamma_k}=\int_{\tau_{k-1}}^{\tau_k}p^T\left(\tau\right)\frac{\partial f}{\partial\gamma_k}\left(\tau,\gamma_k\right)\,d\tau=0, \tag{46}$$

where p is the costate with dynamics

$$\dot{p}=-\left(\frac{\partial F}{\partial x}\right)^T p-\left(\frac{\partial H}{\partial x}\right)^T \tag{47}$$

and boundary condition

$$p\left(T\right)=0. \tag{48}$$

Proof Perturbing the parameter γ_k that defines the kth mode, the dynamics of the perturbed system are

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

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

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

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

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

To derive the optimality conditions, it is necessary to calculate

$$J\left(\gamma_k+\Delta\gamma_k\right)-J\left(\gamma_k\right)=\frac{\partial J}{\partial\gamma_k}\Delta\gamma_k=\Delta J_{\gamma_k}, \tag{49}$$

where

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

simplifies to

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

Defining the costate as

$$p^T\left(\tau\right) = \int_\tau^T \frac{\partial H}{\partial x}\left(t\right) \Phi\left(t, \tau\right) \, dt, \tag{50}$$

the expression for ΔJ_{γ_k} can be rewritten as

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

Seeing that the previous equation matches the form in (49), we finally arrive at the optimality condition for γ_k, given by

$$\frac{\partial J}{\partial \gamma_k} = \int_{\tau_{k-1}}^{\tau_k} p^T\left(\tau\right) \frac{\partial f}{\partial \gamma_k}\left(\tau, \gamma_k\right) \, d\tau. \tag{51}$$

Now, it is necessary to derive an expression for the dynamics and boundary conditions of the costate. Taking the time-derivative of the costate defined in (50) results in

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

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

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

Moving terms out of the integral gives

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

where by reapplying the definition of the costate in (50), we see that the costate evolves as

$$\dot{p} = - \left(\frac{\partial F}{\partial x} \right)^T p - \left(\frac{\partial H}{\partial x} \right)^T. \tag{52}$$

The boundary condition for the costate is found by letting $\tau = T$ in (50), which yields

$$p\left(T\right) = 0. \tag{53}$$

To apply these results to the problem of optimal decentralization, the optimality conditions and costate dynamics need to be specialized for the multi-agent system (40) and cost (41).

Corollary 1 *Optimality conditions for parameters a_k and b_k in the multi-agent system (40), with respect to cost (41), are given by*

$$\frac{\partial J}{\partial a_k} = \int_{\tau_{k-1}}^{\tau_k} p^T(\tau) \frac{\partial f}{\partial a_k}(\tau, a_k)\, d\tau = 0 \tag{54}$$

$$\frac{\partial J}{\partial b_k} = \int_{\tau_{k-1}}^{\tau_k} p^T(\tau) \frac{\partial f}{\partial b_k}(\tau, b_k)\, d\tau = 0, \tag{55}$$

where the $\frac{\partial f}{\partial a_k}$ and $\frac{\partial f}{\partial b_k}$ matrices can be populated using (2×1) blocks based on

$$\frac{\partial f_i}{\partial a_{ijk}} = -\left(x^i - x^j\right) \tag{56}$$

$$\frac{\partial f_i}{\partial b_{ijk}} = -\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\left(x^i - x^j\right), \tag{57}$$

and f_i is the (2×1) block of f corresponding to the dynamics of agent i.

Proof The expressions (54) and (55) are the same as (46) with a_k and b_k substituted in for γ_k. It is unclear whether or not there exists elegant matrix equations to express $\frac{\partial f}{\partial a_k}$ and $\frac{\partial f}{\partial b_k}$. However, some insight can be gained by looking at the individual agents' dynamics (35) with a_{ijk} and b_{ijk} substituted in for M_{ijk}:

$$\dot{x}^i = -\sum_{j \in N(i)} \begin{bmatrix} a_{ijk} & -b_{ijk} \\ b_{ijk} & a_{ijk} \end{bmatrix}\left(x^i - x^j\right). \tag{58}$$

From the above expression, we see that a_{ijk} and b_{ijk} do not appear anywhere else except in agent i's dynamics. Therefore, it can be differentiated by a_{ijk} and b_{ijk} to obtain the expressions for $\frac{\partial f_i}{\partial a_{ijk}}$ and $\frac{\partial f_i}{\partial b_{ijk}}$ in (56) and (57), respectively. Those results can then be put composed together to obtain $\frac{\partial f}{\partial a_k}$ and $\frac{\partial f}{\partial b_k}$.

Finally, the multi-agent system's dynamics (40) and cost functional (41) need to be substituted into the costate dynamics (47).

Corollary 2 *Costate dynamics for calculating the optimality conditions of a_k and b_k in (54) and (55) are*

$$\dot{p}(t) = L_k^T(a_k, b_k) p(t) - x(t) + x_d(t) \quad \forall t \in [\tau_{k-1}, \tau_k). \tag{59}$$

5.3 Switch Time Optimization

Optimality conditions for switching times in a switched autonomous system were derived in Egerstedt et al (2006). They are restated here, for the sake of easy reference:

Theorem 4 *The optimality condition with respect to cost functional (45) for switching times in a switched autonomous system, where mode k has dynamics f parameterized by γ_k, is*

$$\frac{\partial J}{\partial \tau_k} = p^T(\tau_k)\left(f(x(\tau_k), \gamma_k) - f(x(\tau_k), \gamma_{k+1})\right) = 0. \tag{60}$$

The costate dynamics are the same as (47), with associated boundary conditions (48).

Corollary 3 *The switch time optimality conditions specialized for the multi-agent system (40), with respect to cost (41), are given by*

$$\frac{\partial J}{\partial \tau_k} = p^T(\tau_k)\left(L_{k+1}(a_{k+1}, b_{k+1}) - L_k(a_k, b_k)\right) x(\tau_k), \tag{61}$$

with costate dynamics (59) and boundary condition (48).

5.4 Generating a Locally Executable GPS

The mode parameters a_k and b_k, as well as the global switching times τ_k, can be numerically optimized with respect to the cost J in (41) using the derived costate dynamics and optimality conditions. One such optimization technique, which is used in the example at the end of this section, is the steepest descent with Armijo step size algorithm. However, as stated in Egerstedt et al (2006), additional care must be taken whenever global switching times are being optimized to ensure that they satisfy the constraint (30).

From this point onwards, we assume that given some desired agent trajectory, a numerical optimization algorithm was used to find the mode parameters a_k^* and b_k^*, and global switch times τ_k^*, that yielded a tolerable final cost J. These optimized values help define a set of decentralized controllers for our multi-agent system to track the desired motion with. We will now show how to encapsulate each of the generated decentralized controllers into decentralized consistent atoms. GPS will be used to script a mode sequence for the multi-agent system that goes through the controllers sequentially, and switches based on timer-based conditions. It will be shown later that a GPS control sequence scripted using this method is always locally executable.

Define the set containing the graph inducing function s_σ as

$$\mathbb{S}_\sigma = \{s_\sigma\}. \tag{62}$$

Moreover, let the decentralized multi-agent controller corresponding to the kth mode be given by $\mathcal{U}_k^*(s,x,t)$, where

$$\mathcal{U}_k^{*i}(s,x,t) = \begin{cases} -\sum\limits_{j\in N(i)} r_{ijk}^* \, Rot\left(\theta_{ijk}^*\right)\left(x^i - x^j\right) \text{ , for } t < T_0 + \left(\tau_k^* - \tau_{k-1}^*\right) \\ 0 \qquad\qquad\qquad\qquad\qquad \text{, otherwise,} \end{cases} \tag{63}$$

is simply the decentralized controller from (31) but using the optimal parameters r_{ijk}^* and θ_{ijk}^*, as obtained from a_k^* and b_k^*. In the above expression, T_0 corresponds to the time at which that particular controller is first executed by agents in the system. The reason for limiting the duration of each controller is because they are time-invariant, and that agents have single integrator dynamics. Therefore, arbitrarily long "breaks" can be taken in between executing the controllers for each mode, as long as the duration that each controller is executed matches what is specified by the optimized global switching times. Note that this controller assumes that agents have access to an accurate clock. Such an assumption was also used in the examples in Section 4, where agents used timer interrupts for switching between line and circle formation atoms.

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

$$\mathcal{C}_k^{*i}(s,x,t) = \begin{cases} 1 \text{ , if } t \geq T_0 + \left(\tau_k^* - \tau_{k-1}^*\right) \\ 0 \text{ , otherwise.} \end{cases} \tag{64}$$

Finally, we define the set of initial graphs $\mathbb{G}_k^*$ and final graphs $\mathbb{H}_k^*$. These sets will contain the information flow graphs corresponding to the optimized state trajectory at the switching times before and after the execution of each controller. Let $x^* : [0,T] \to \mathbb{R}^{2N}$ be the state trajectory resulting from using the optimized mode parameters a_k^* and b_k^*, as well as the optimized global switching times τ_k^*, on the multi-agent system dynamics (40). The set of initial and final graphs are then given by

$$\mathbb{G}_k^* = \left\{ s_\sigma(x^*(\tau_{k-1}^*)) \right\} \text{ and } \mathbb{H}_k^* = \left\{ s_\sigma(x^*(\tau_k^*)) \right\}. \tag{65}$$

Putting all the components together, a decentralized consistent atom can be constructed to encapsulate the optimized decentralized controller for each mode $k = 1, \ldots, m$.

Lemma 5 *The atom for the kth decentralized controller resulting from optimal decentralization:*

$$\mathcal{A}_k^* = (\mathbb{S}_\sigma, \mathbb{F}_\mathcal{I}, \mathbb{G}_k^*, \mathbb{H}_k^*, \mathcal{U}_k^*, \mathcal{C}_k^*), \tag{66}$$

for $k = 1, \ldots, m$, is a decentralized consistent atom.

Proof The set $\mathbb{G}_k^*$ contains only the information flow graph resulting from applying the graph inducing function s_σ to the agent state vector $x^*(\tau_{k-1}^*) + \alpha$, where $\alpha \in \mathbb{R}^{2N}$ is any arbitrary vector. This is because $s_\sigma(x^*(\tau_{k-1}^*) + \alpha) = s_\sigma(x^*(\tau_{k-1}^*)) \in \mathbb{G}_k^*$. Since $\mathcal{U}_k^*$ uses only relative displacement vectors,

we know that the agent state vector resulting from executing the control law with dynamics $f_{\mathcal{I}}$ for at least $\tau_k^* - \tau_{k-1}^*$ time units will be $x^*(\tau_k^*) + \alpha$ (since the control is designed to be zero after $\tau_k^* - \tau_{k-1}^*$ seconds have elapsed). The function $\mathcal{C}_k^*$ allows for each agent to individually determine when the controller has executed for long enough by using a timer interrupt. The information flow graph of the final state is then given by $s_\sigma(x^*(\tau_k^*) + \alpha) = s_\sigma(x^*(\tau_k^*)) \in \mathbb{H}_k^*$.

To script a control sequence using the decentralized consistent atoms, they must first be placed into decentralized modes. For the sake of analysis, we will let the interrupt function ξ_k^* for mode k be such that $\xi_k^* \to 1$ always. However, recall that the interrupt functions ξ_k^* can be defined arbitrarily in implementation since they do not affect the composability of modes. The kth mode in the sequence is therefore given by

$$\mathcal{M}_k^* = (\mathcal{A}_k^*, \xi_k^*)\,, \tag{67}$$

for $k = 1, \ldots, m$. With all m modes defined, a locally executable GPS can be constructed that contains a description of our multi-agent system, as well as a mode sequence for the agents to execute that makes them sequentially transition through the generated decentralized controllers.

Theorem 5 *The graph process specification resulting from performing optimal decentralization:*

$$GPS^* = ((x_0, s_\sigma, f_{\mathcal{I}}), (\mathcal{M}_1^*, \ldots, \mathcal{M}_m^*)), \tag{68}$$

is locally executable.

Proof The graph inducing function s_σ and dynamics $f_{\mathcal{I}}$ are members of their associated sets in each mode's decentralized consistent atom. Using the graph inducing function on the initial condition gives $s_\sigma(x_0) \in \mathbb{G}_1^*$. Furthermore, since $\mathbb{H}_k^* = \mathbb{G}_{k+1}^*$, we get that $\mathcal{M}_k^* \prec \mathcal{M}_{k+1}^*$, for $k = 1, \ldots, m-1$.

Therefore, what has been shown is that decentralized controller sequences generated from optimal decentralization can always be encapsulated into decentralized consistent atoms. Those atoms can then be used to script a locally executable GPS for agents to track desired multi-agent motions with. We will now showcase the performance of decentralized control strategies scripted using this method through a simulation of agents tracking a drumline-inspired multi-agent dance.

Example 2 A MATLAB simulation was performed in which a system of $N = 21$ agents were tasked to track the drumline-inspired multi-agent dance consisting of agent trajectories shown in Figure 1. Drumline formations are traditionally designed by choreographers to be executed in a centralized manner. The position and path taken by band members at each moment in time have been predetermined to a high level of detail. As a result, band members spend a lot of time practicing to follow these predetermined paths. However, such an approach requires each band member to memorize paths taken throughout the

entire dance sequence and have global sensing capabilities to know if they're in the correct position. Optimal decentralization is used to mimic the original routine using only decentralized control laws, i.e., requiring agents to make use of only locally-available information while executing the control laws.

The target trajectory is defined from $t = 0$ to $t = T = 10.78$. A total of $m = 23$ modes were allowed for the multi-agent system to track the desired multi-agent motion. Therefore, all 22 switching times, as well as the parameters for all 23 modes, need to be optimized. To do so, steepest descent with Armijo step size algorithm was performed for 5000 iterations. The resulting convergence of the cost J, corresponding to the tracking error, is shown in Figure 6. Using the decentralized control laws and switching times generated from optimal decentralization, decentralized consistent atoms and interrupt conditions can be created using methods shown in Section 5.4. Putting them together, a locally executable GPS can be constructed for the multi-agent system that makes agents transition through a sequence of decentralized control laws with timer-based interrupts to track the drumline-inspired dance.

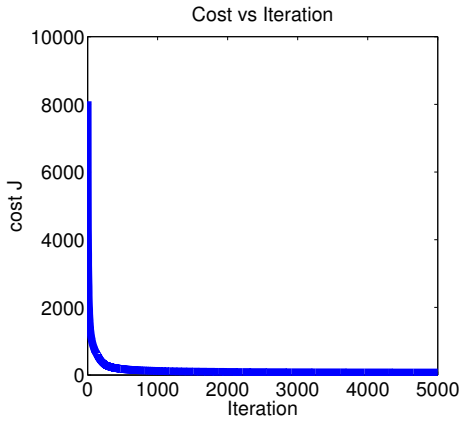


Fig. 6 Convergence of cost J (tracking error) after performing steepest descent with Armijo step size on parameters and switching times for tracking a multi-agent drumline-inspired dance.

Note that everything in this example up until now is meant to be done offline and in a centralized manner. Upon scripting the decentralized control sequence using GPS and checking to see that it is indeed locally executable, the information required for execution (i.e., control laws, termination condition checks, and interrupt conditions) are then downloaded to each agent. The agents then simultaneously start executing their respective control strategies using only information that is locally available in the network.

The agent trajectories resulting from a simulation where they execute the scripted decentralized control strategy for tracking the drumline-inspired dance are shown in Figure 7. Here, the actual locations of the agents are marked by O's with lines connecting them to their desired location marked

by X's to help ease the comparison. Note that while the original simulation of the multi-agent dance in Figure 1 used performed using centralized coordination strategies, Figure 7 shows agents tracking the trajectories but using only decentralized controllers. Therefore, this example helps showcase the performance of the optimal decentralization algorithm in generating controllers for agents to track complex multi-agent motions.

6 Conclusion

This paper presented an extension to previous work on the Graph Process Specification (GPS) framework, as well as a series of detailed examples demonstrating its usage in multi-agent control design. It was shown that multi-agent controllers oftentimes have network topological prerequisites which must be respected, while being executed by agents, in order to have the desired effect on a system. The concept behind atoms, the fundamental building blocks in GPS, was presented along with examples on how to construct them. Each atom specified a specific network topological transition for the system. Moreover, each atom also described the means by which agents can make this transition occur, as well as a way for agents to locally detect that the transition has taken place. Using a library of atoms, GPS could be used to script decentralized control sequences for agents, while ensuring that the network topological requirements of each decentralized controller were satisfied. This paper also presented an example of how an atom library can be constructed. The optimal decentralization algorithm was used to generate atoms for agents to track desired multi-agent motions with in a system where the network topology is static. The paper concluded with a simulation showcasing agents performing a complex drumline-inspired dance using decentralized control sequences generated from optimal decentralization and scripted using GPS.

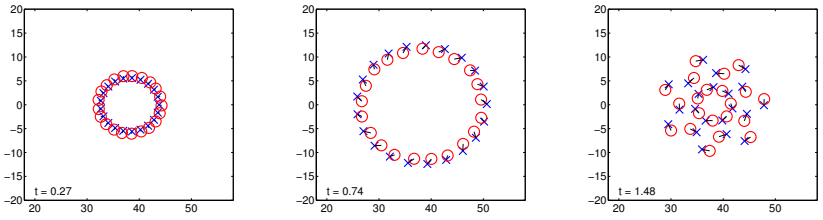
Acknowledgements This work was sponsored by the US National Science Foundation through Grant # CCF 0820004, and ONR through MURI HUNT. The authors would also like to thank Prof. Christopher Moore for discussions about the Georgia Tech drumline.

References

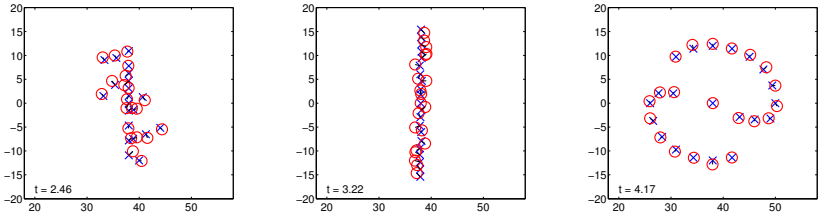
Alamir M, Attia S (2004) On solving optimal control problems for switched hybrid nonlinear systems by strong variations algorithms. In: Proceedings of 6th IFAC symposium on nonlinear control systems, pp 558–563

Antsaklis P (2000) A brief introduction to the theory and applications of hybrid systems. In: Proc IEEE, Special Issue on Hybrid Systems: Theory and Applications, Citeseer

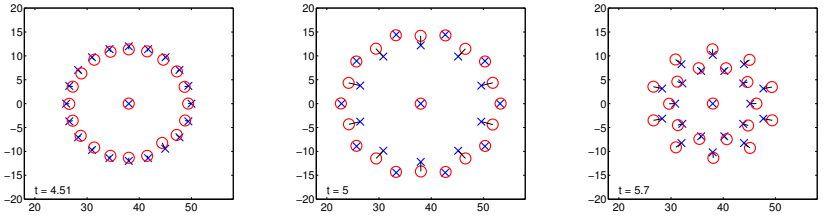
Attia S, Alamir M, de Wit C (2005) Sub optimal control of switched nonlinear systems under location and switching constraints. In: IFAC World Congress



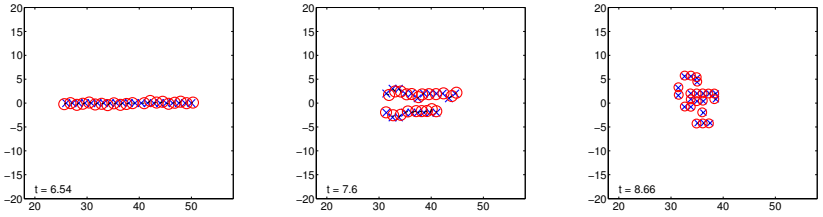
(a) Circle expands outwards. (b) Circle spirals inwards. (c) Circles rotate in different directions.



(d) Circles break to form a line. (e) Agents form vertical line. (f) Line unravels to circle with center.



(g) Agents form a circle with center. (h) Certain agents expand outwards. (i) Formation shrinks towards center.



(j) Agents form a horizontal line. (k) Agents split into two groups. (l) Agents form a GT (Georgia Tech) logo.

Fig. 7 Simulation of $N = 21$ agents executing the drumline-inspired dance from Figure 1 using a decentralized controller sequence generated by the optimal decentralization algorithm and scripted using GPS. The locations of the agents are marked by O's with lines connecting them to their desired location marked by X's.

Axelsson H, Wardi Y, Egerstedt M, Verriest E (2008) Gradient descent approach to optimal mode scheduling in hybrid dynamical systems. Journal of Optimization Theory and Applications 136(2):167–186

Bamieh B, Paganini F, Dahleh M (2002) Distributed control of spatially invariant systems. Automatic Control, IEEE Transactions on 47(7):1091–1107

Bemporad A, Borrelli F, Morari M (2000) Piecewise linear optimal controllers for hybrid systems. In: American Control Conference, 2000. Proceedings of the 2000, IEEE, vol 2, pp 1190–1194

Branicky M, Borkar V, Mitter S (1998) A unified framework for hybrid control: Model and optimal control theory. Automatic Control, IEEE Transactions on 43(1):31–45

Brockett R (1988) On the computer control of movement. In: Proceedings of IEEE International Conference on Robotics and Automation, vol 1, pp 534–540

Couzin I, Franks N (2003) Self-organized lane formation and optimized traffic flow in army ants. Proceedings of the Royal Society of London Series B: Biological Sciences 270(1511):139

Egerstedt M, Wardi Y, Axelsson H (2006) Transition-time optimization for switched-mode dynamical systems. Automatic Control, IEEE Transactions on 51(1):110–115

Eren T, Whiteley W, Anderson B, Morse A, Belhumeur P (2005) Information structures to secure control of rigid formations with leader-follower architecture. In: American Control Conference, 2005. Proceedings of the 2005, IEEE, pp 2966–2971

Hedlund S, Rantzer A (1999) Optimal control of hybrid systems. In: Decision and Control, 1999. Proceedings of the 38th IEEE Conference on, IEEE, vol 4, pp 3972–3977

Jadbabaie A, Lin J, Morse A (2003) Coordination of groups of mobile autonomous agents using nearest neighbor rules. Automatic Control, IEEE Transactions on 48(6):988–1001

Ji M, Egerstedt M (2007) Distributed coordination control of multiagent systems while preserving connectedness. IEEE Transactions on Robotics 23(4):693–703

Johansson K, Egerstedt M, Lygeros J, Sastry S (1999) On the regularization of Zeno hybrid automata. Systems and Control Letters 38(3):141–150

Kloetzer M, Belta C (2007) Temporal logic planning and control of robotic swarms by hierarchical abstractions. IEEE Transactions on Robotics 23(2):320–330

Koutsoukos X, Antsaklis P, Stiver J, Lemmon M (2000) Supervisory control of hybrid systems. Proceedings of the IEEE 88(7):1026–1049

Lin Z, Broucke M, Francis B (2004) Local control strategies for groups of mobile autonomous agents. Automatic Control, IEEE Transactions on 49(4):622–629

Liu Y, Passino K, Polycarpou M (2003) Stability analysis of one-dimensional asynchronous swarms. Automatic Control, IEEE Transactions on 48(10):1848–1854

Lynch N (1996) Distributed algorithms. Morgan Kaufmann
Manikonda V, Krishnaprasad PS, Hendler J (1998) Languages, behaviors, hybrid architectures and motion control. In: Willems J, Baillieul J (eds) Mathematical Control Theory, Springer-Verlag
Martin P, de la Croix J, Egerstedt M (2008) MDLn: A motion description language for networked systems. In: Proceedings of 47th IEEE Conference on Decision and Control
McNew J, Klavins E (2006) Locally interacting hybrid systems with embedded graph grammars. In: Decision and Control, 2006 45th IEEE Conference on, pp 6080–6087
Mesbahi M, Egerstedt M (2010) Graph Theoretic Methods for Multiagent Networks. Princeton University Press, Princeton, NJ
Morse A (1997) Control using logic-based switching. Citeseer
Motee N, Jadbabaie A (2008) Optimal control of spatially distributed systems. Automatic Control, IEEE Transactions on 53(7):1616–1629
Muhammad A, Egerstedt M (2005) Connectivity graphs as models of local interactions. Applied Mathematics and Computation 168(1):243–269
Olfati-Saber R, Murray R (2004) Consensus problems in networks of agents with switching topology and time-delays. Automatic Control, IEEE Transactions on 49(9):1520–1533
Olfati-Saber R, Fax J, Murray R (2007) Consensus and cooperation in networked multi-agent systems. Proceedings of the IEEE 95(1):215–233
Rantzer A (2007) A separation principle for distributed control. In: Decision and Control, 2006 45th IEEE Conference on, IEEE, pp 3609–3613
Ren W, Beard R (2005) Consensus seeking in multiagent systems under dynamically changing interaction topologies. Automatic Control, IEEE Transactions on 50(5):655–661
Rotkowitz M, Lall S (2006) A characterization of convex problems in decentralized Control. Automatic Control, IEEE Transactions on 51(2):274–286
Shaikh M, Caines P (2002) On trajectory optimization for hybrid systems: Theory and algorithms for fixed schedules. In: IEEE Conference on Decision and Control, IEEE; 1998, vol 2, pp 1997–1998
Shaikh M, Caines P (2003) On the optimal control of hybrid systems: Optimization of trajectories, switching times, and location schedules. In: Proceedings of the 6th international conference on Hybrid systems: computation and control, Springer-Verlag, pp 466–481
Shaikh M, Caines P (2007) On the hybrid optimal control problem: Theory and algorithms. Automatic Control, IEEE Transactions on 52(9):1587–1603
Smith B, Howard A, McNew J, Wang J, Egerstedt M (2009) Multi-robot deployment and coordination with Embedded Graph Grammars. Autonomous Robots 26(1):79–98
Tanner H, Pappas G, Kumar V (2004) Leader-to-formation stability. Robotics and Automation, IEEE Transactions on 20(3):443–455
Tanner H, Jadbabaie A, Pappas G (2007) Flocking in fixed and switching networks. Automatic Control, IEEE Transactions on 52(5):863–868

Twu P, Egerstedt M (2010) Optimal decentralization of multi-agent motions. In: American Control Conference (ACC), 2010, IEEE, pp 2326–2331

Twu P, Martin P, Egerstedt M (2010) Graph process specifications for hybrid networked systems. In: Proceedings of 10th International Workshop on Discrete Event Systems

Xiao L, Boyd S, Lall S (2006) A space-time diffusion scheme for peer-to-peer least-squares estimation. In: Proceedings of the 5th international conference on Information processing in sensor networks, ACM, pp 168–176

Xu X, Antsaklis P (2002a) Optimal control of switched autonomous systems. In: Decision and Control, 2002, Proceedings of the 41st IEEE Conference on, IEEE, vol 4, pp 4401–4406

Xu X, Antsaklis P (2002b) Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions. International Journal of Control, 75 16(17):1406–1426