Mean-Field Stabilization of Markov Chain Models for Robotic Swarms: Computational Approaches and Experimental Results

Vaibhav Deshmukh, Karthik Elamvazhuthi, Shiba Biswal, Zahi Kakish, and Spring Berman

***Abstract*—In this paper, we present two computational approaches for synthesizing decentralized density-feedback laws that asymptotically stabilize a strictly positive target equilibrium distribution of a swarm of agents among a set of states. The agents' states evolve according to a continuous-time Markov chain on a bidirected graph, and the density-feedback laws are designed to prevent the agents from switching between states at equilibrium. First, we use classical Linear Matrix Inequality (LMI)-based tools to synthesize linear feedback laws that (locally) exponentially stabilize the desired equilibrium distribution of the corresponding mean-field model. Since these feedback laws violate positivity constraints on the control inputs, we construct rational feedback laws that respect these constraints and have the same stabilizing properties as the original feedback laws. Next, we present a Sum-of-Squares (SOS)-based approach to constructing polynomial feedback laws that globally stabilize an equilibrium distribution and also satisfy the positivity constraints. We validate the effectiveness of these control laws through numerical simulations with different agent populations and graph sizes and through multi-robot experiments on spatial redistribution among four regions.**

***Index Terms*—Swarms, Multi-Robot Systems, Distributed Robot Systems, Optimization and Optimal Control, Probability and Statistical Methods**

I. INTRODUCTION

IN recent years, there has been much work on developing mean-field models of robotic swarms with stochastic behaviors, e.g. [2], [14], [18], [5], [16], [12], [1], and using these models to design swarm control strategies that are scalable with the number of robots and robust to robot failures. In this paper, we consider such an approach for controlling a swarm of robots to allocate among a set of states in a decentralized fashion, that is, using only information that the robots can obtain from their local environment. We model the time evolution of each robot's state as a continuous-time Markov chain (CTMC) and frame the control problem in terms of the *forward equation*, or mean-field model, of the system. Our goal is to algorithmically construct identity-invariant robot control laws that guarantee stabilization of the swarm to a target state distribution.

In existing approaches to this problem that use Markov chain models [5], [1], the robots continue switching between states at equilibrium, which could unnecessarily expend energy, even though the swarm distribution among states in the mean-field model is stabilized. This continued switching at equilibrium is due to the time- and density-independence of the control laws considered in these works. To address this problem, [11] constructed density-feedback laws that use *quorum-sensing strategies* to stabilize a swarm to a desired distribution and reduce the amount of switching between states at equilibrium. In [17], the authors pose this as a problem of controlling the variance of the agent distribution at equilibrium using density-feedback laws. For the case where the agents' states evolve according to a discrete-time Markov chain (DTMC), this problem has been addressed in [3], [7], [4] by allowing the control laws to depend on time.

Recently, we considered this problem for the case where agents' states evolve according to CTMCs [10]. As in [17], we proposed density-feedback laws that stabilize a swarm of agents to a strictly positive target distribution. However, differently from [17], we considered feedback laws that are linear or polynomial functions of the state; that do not violate positivity constraints; and that converge to zero at equilibrium, preventing further switching between states. We constructively proved the existence of these stabilizing linear and polynomial control laws by presenting two specific examples of such control laws.

The two feedback controllers that we proposed in [10] could yield poor system performance in practice. In this paper, we develop a principled approach to designing density-feedback controllers with desired performance characteristics. Toward this end, we present algorithmic procedures for constructing linear and polynomial feedback controllers that stabilize the mean-field model from [10] to a strictly positive target distribution with no state transitions occurring at equilibrium. These procedures can incorporate additional constraints to improve the properties of the closed-loop system response. The first procedure adapts Linear Matrix Inequality (LMI)-based tools from linear systems theory to synthesize locally exponentially stabilizing rational feedback laws from linear control laws. Since these control laws are only locally stabilizing and can take unbounded values away from the equilibrium, we also develop a second procedure for control synthesis using Sum-of-Squares (SOS)-based techniques. Using the MATLAB toolbox

Manuscript received: September 10, 2017; Revised November 21, 2017; Accepted December 13, 2017.

This paper was recommended for publication by Editor Nak Young Chong upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by ONR Young Investigator Award N00014-16-1-2605 and by the Arizona State University Global Security Initiative.

The authors are with the School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ, 85281 USA {vdeshmuk, karthikevaz, sbiswal, zahi.kakish, spring.berman}@asu.edu.

Digital Object Identifier (DOI): see top of this page.

SOSTOOLS [19], we construct nonlinear polynomial control laws that are globally asymptotically stabilizing. Moreover, in contrast to the control approaches presented in [4], [7], all computations for control synthesis are done offline in our procedures rather than onboard the robots in real-time, which reduces the robots' computational burden. We validate both the linear and nonlinear feedback laws through numerical simulations and through experiments with mobile robots that redistribute themselves among four spatial regions.

II. PROBLEM FORMULATION

A. Notation

We denote by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ a directed graph with M vertices, $\mathcal{V} = \{1, 2, ..., M\}$, and a set of $N_\mathcal{E}$ edges, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. We say that $e = (i, j) \in \mathcal{E}$ if there is an edge from vertex $i \in \mathcal{V}$ to vertex $j \in \mathcal{V}$. We define a source map $S : \mathcal{E} \to \mathcal{V}$ and a target map $T : \mathcal{E} \to \mathcal{V}$ for which $S(e) = i$ and $T(e) = j$ whenever $e = (i, j) \in \mathcal{E}$. We assume that $(i, i) \notin \mathcal{E}$ for all $i \in \mathcal{V}$. The graph $\mathcal{G}$ is said to be *bidirected* if $e = (S(e), T(e)) \in \mathcal{E}$ implies that $\tilde{e} = (T(e), S(e))$ also lies in $\mathcal{E}$. In this paper, we will only consider bidirected graphs.

B. Problem Statement

Consider a swarm of N autonomous agents whose states evolve in continuous time according to a Markov chain with finite state space $\mathcal{V}$. As an example application, $\mathcal{V}$ can represent a set of spatial locations that are obtained by partitioning the agents' environment. The graph $\mathcal{G}$ determines the pairs of vertices (states) between which the agents can transition. We define $u_e : [0, \infty) \to \mathbb{R}_+$ as a *transition rate* for each $e = (i, j) \in \mathcal{E}$, where $\mathbb{R}_+$ is the set of positive real numbers. The evolution of the N agents' states over time t on the state space $\mathcal{V}$ is described by N stochastic processes, $X_k(t) \in \mathcal{V}$, $k = 1, ..., N$. Each stochastic process $X_k(t)$ evolves according to the following conditional probabilities for every $e \in \mathcal{E}$:

$$\mathbb{P}(X_k(t + h) = T(e)|X_k(t) = S(e)) = u_e(t)h + o(h). \quad (1)$$

Here, $o(h)$ is the little-oh symbol and $\mathbb{P}$ is the underlying probability measure defined on the space of events Ω (which will be left undefined, as is common) that is induced by the stochastic processes $\{X_k(t)\}_{k=1}^N$. Let $\mathcal{P}(\mathcal{V})$ be the $(M-1)$-dimensional simplex of probability densities on $\mathcal{V}$, defined as $\mathcal{P}(\mathcal{V}) = \{\mathbf{y} \in \mathbb{R}_+^M : \sum_i y_i = 1\}$. Let $\mathbf{x}(t) = [x_1(t) \ ... \ x_M(t)]^T \in \mathcal{P}(\mathcal{V})$ be the vector of probability distributions of the random variable $X_k(t)$ at time t, that is,

$$x_i(t) = \mathbb{P}(X_k(t) = i), \quad i \in \mathcal{V}. \quad (2)$$

The evolution of probability distributions is determined by the *Kolmogorov forward equation*, which can be cast in an explicitly control-theoretic form as a bilinear control system,

$$\dot{\mathbf{x}}(t) = \sum_{e \in \mathcal{E}} u_e(t)\mathbf{B}_e\mathbf{x}(t), \quad \mathbf{x}(0) = \mathbf{x}^0 \in \mathcal{P}(\mathcal{V}), \quad (3)$$

where $\mathbf{B}_e$, $e \in \mathcal{E}$, are control matrices with entries

$$B_e^{ij} = \begin{cases} -1 & \text{if } i = j = S(e), \\ 1 & \text{if } i = T(e),\ j = S(e), \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Here, B_e^{ij} denotes the element in row i and column j of the matrix $\mathbf{B}_e$.

Using these definitions, we can now state the main problem addressed in this paper.

Problem II.1. *Given a strictly positive desired equilibrium distribution $\mathbf{x}^{eq} \in \mathcal{P}(\mathcal{V})$, compute transition rates $k_e : \mathcal{P}(\mathcal{V}) \to \mathbb{R}_+$, $e \in \mathcal{E}$, such that the closed-loop system*

$$\dot{\mathbf{x}}(t) = \sum_{e \in \mathcal{E}} k_e(\mathbf{x}(t))\mathbf{B}_e\mathbf{x}(t) \quad (5)$$

satisfies $\lim_{t\to\infty} \|\mathbf{x}(t) - \mathbf{x}^{eq}\| = 0$ for all $\mathbf{x}^0 \in \mathcal{P}(\mathcal{V})$, with the additional constraint that $k_e(\mathbf{x}^{eq}) = 0$ for all $e \in \mathcal{E}$. Moreover, the density feedback should have a decentralized structure, in that each k_e must be a function only of densities x_i for which $i = S(e)$ or $i = S(\tilde{e})$, where $T(\tilde{e}) = S(e)$.

We specify that each agent knows the desired equilibrium distribution $\mathbf{x}^{eq}$. This assumption is used in other approaches to stabilizing solutions of the mean-field model of a swarm to desired probability distributions, e.g. [1], [5], [11], [17].

III. CONTROLLER DESIGN

In this section, we present two algorithmic approaches to obtaining sets of linear and nonlinear feedback control laws $k_e(\mathbf{x})$ that solve Problem II.1. In Section III-A, we adapt Linear Matrix Inequality (LMI) techniques from linear system theory to our problem in order to compute decentralized linear control laws. In Section III-B, we present a Sum-of-Squares (SOS) based approach for the synthesis of decentralized nonlinear control laws.

A. Linear LMI-Based Controller

To synthesize linear controllers, we use a linearization-based approach for controller design. The control system (3) is linearized about the desired equilibrium state $\mathbf{x}^{eq}$ and the equilibrium control inputs $k_e(\mathbf{x}^{eq})$. Since we require the agents to stop switching between states at equilibrium, we set the equilibrium control inputs to be $k_e(\mathbf{x}^{eq}) = 0$ for each $e \in \mathcal{E}$. Let $\mathcal{I} : \mathcal{E} \to \{1, 2, ..., N_\mathcal{E}\}$ be a bijective map, i.e., an ordering of $\mathcal{E}$, and let $\delta u_{\mathcal{I}(e)} = u_e - k_e(\mathbf{x}^{eq}) = u_e$ for each $e \in \mathcal{E}$. Defining $\delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}^{eq}$ and $\delta\mathbf{u}(t) = [\delta u_1(t) \ ... \ \delta u_{N_\mathcal{E}}(t)]^T$, we obtain the linearized control system

$$\delta\dot{\mathbf{x}}(t) = \tilde{\mathbf{B}}\delta\mathbf{u}(t), \quad \delta\mathbf{x}(0) = \mathbf{x}^0 - \mathbf{x}^{eq}, \quad (6)$$

in which column $i = \mathcal{I}(e)$ of the matrix $\tilde{\mathbf{B}}$ is equal to $\tilde{B}_i = \mathbf{B}_e\mathbf{x}^{eq}$. To ensure that system (3) is sufficiently stabilizable, we assume that the equilibrium state is strictly positive, i.e. $x_i^{eq} > 0$ for each $i \in \mathcal{V}$. If the graph $\mathcal{G}$ is strongly connected, then it follows from Perron-Frobenius theorem that the system (6) has $M-1$ controllable eigenvalues and a single uncontrollable eigenvalue at 0. This is true because $\mathbf{x}^{eq}$ is assumed to be have all positive elements, and hence $\tilde{\mathbf{B}}$ has rank $M-1$. The eigenvalue at 0 is uncontrollable because the simplex $\mathcal{P}(V)$ is invariant for the system (5). This follows from the fact that each column of $\mathbf{B}_e$ sums to 0, which implies that the sum $\sum_{i\in\mathcal{V}} x_i(t)$ remains constant for all $t \geq 0$. Moreover, the off-diagonal terms of $\mathbf{B}_e$ are non-negative, and hence $x_i(t) \geq 0$ for all $t \geq 0$ and all $i \in \mathcal{V}$.

We now discuss the algorithmic construction of decentralized linear control laws that exponentially stabilize the linearized system (6), and therefore locally stabilize the original system (5). The linear control laws $k_e(\mathbf{x})$ that we construct will always violate the positivity constraints at some point in $\mathcal{P}(\mathcal{V})$. To see this explicitly, suppose that $\boldsymbol{\epsilon} = [\epsilon_1 \ldots \epsilon_M]^T \in \mathbb{R}^M$ is a nonzero element such that $\mathbf{x}^{eq} \pm \boldsymbol{\epsilon} \in \mathcal{P}(\mathcal{V})$, and suppose that $k_e(\mathbf{x})$ is a linear control law. Then the control law has the form $k_e(\mathbf{x}) = \sum_{i\in\mathcal{V}} a_e^i x_i + b_e$, where a_e^i and b_e are gain parameters. Since the control inputs must take the value 0 at equilibrium, we must have that $b_e = -\sum_{i\in\mathcal{V}} a_e^i x_i^{eq}$. Suppose, for the sake of contradiction, that this linear control law satisfies the positivity constraints; that is, the range of $k_e(\mathbf{x})$ is $[0,\infty)$ for some $e \in \mathcal{E}$. Then we must have that $k_e(\mathbf{x}^{eq} + \boldsymbol{\epsilon}) = \sum_{i\in\mathcal{V}} a_e^i (x_i^{eq} + \epsilon_i) + b_e = \sum_{i\in\mathcal{V}} a_e^i \epsilon_i > 0$. This must imply that $k_e(\mathbf{x}^{eq} - \boldsymbol{\epsilon}) = \sum_{i\in\mathcal{V}} a_e^i (x_i^{eq} - \epsilon_i) + b_e = -\sum_{i\in\mathcal{V}} a_e^i \epsilon_i < 0$, which contradicts the original assumption that the control law $k_e(\mathbf{x})$ satisfies the positivity constraints. Hence, to ensure that the control laws satisfy the positivity constraints, we replace them with rational feedback control laws $c_e(\mathbf{x})$ that produce the same closed-loop system trajectories but respect the positivity constraints, as desired. We show that this is possible in the following theorem from our prior work [10].

Theorem III.1. *[10] Let $\mathcal{G}$ be a bidirected graph. Let $k_e : \mathbb{R}^M \to (-\infty,\infty)$ be a map for each $e \in \mathcal{E}$ such that there exists a unique global solution of the system* (5)*. Additionally, assume that $\mathbf{x}(t) \in \operatorname{int}(\mathcal{P}(\mathcal{V}))$ for each $t \in [0,\infty)$. Consider the functions $m_e^p : \mathbb{R}^M \to \{0,1\}$ and $m_e^n : \mathbb{R}^M \to \{0,1\}$, defined for each $e \in \mathcal{E}$ and for all $\mathbf{y} \in \mathbb{R}^M$ as*

$$\begin{aligned} m_e^p(\mathbf{y}) &= 1 \quad \text{if} \quad k_e(\mathbf{y}) \geq 0, \quad 0 \ \text{otherwise}; \\ m_e^n(\mathbf{y}) &= 1 \quad \text{if} \quad k_e(\mathbf{y}) \leq 0, \quad 0 \ \text{otherwise}. \end{aligned} \tag{7}$$

Let $c_e : \mathbb{R}^M \to [0,\infty)$ be given by

$$c_e(\mathbf{y}) = m_e^p(\mathbf{y}) k_e(\mathbf{y}) - m_{\tilde{e}}^n(\mathbf{y}) k_{\tilde{e}}(\mathbf{y}) \frac{y_{T(e)}}{y_{S(e)}}. \tag{8}$$

Then the solution $\tilde{\mathbf{x}}(t)$ of the following system,

$$\begin{aligned} \dot{\tilde{\mathbf{x}}} &= \sum_{e\in\mathcal{E}} c_e(\tilde{\mathbf{x}}(t)) \mathbf{B}_e \tilde{\mathbf{x}}(t), \quad t \in [0,\infty) \\ \tilde{\mathbf{x}}(0) &= \mathbf{x}^0 \in \mathcal{P}(\mathcal{V}), \end{aligned} \tag{9}$$

is unique, defined globally, and satisfies $\tilde{\mathbf{x}}(t) = \mathbf{x}(t)$ for all $t \in [0,\infty)$.

The following result [9] is widely used in the literature on LMI-based tools for construction of linear control laws:

Theorem III.2. *Let $\mathbf{A} \in \mathbb{R}^{M\times M}$ and $\mathbf{B} \in \mathbb{R}^{M\times N_{\mathcal{E}}}$, where $N_{\mathcal{E}}$ is the number of control inputs. Consider the linear control system*

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t). \tag{10}$$

Then a static linear state feedback law, $\mathbf{u} = -\mathbf{K}\mathbf{x}$, stabilizes the system (10) *if and only if there exist matrices $\mathbf{P} > 0$ and $\mathbf{Z}$ such that*

$$\mathbf{K} = \mathbf{Z}\mathbf{P}^{-1}, \tag{11}$$

$$\mathbf{P}\mathbf{A} + \mathbf{B}\mathbf{Z} + \mathbf{P}\mathbf{A}^T + \mathbf{Z}^T\mathbf{B}^T < 0. \tag{12}$$

This result can be used to construct state feedback laws for a general linear system. The theorem is attractive from a computational point of view since the constraints, $\mathbf{P} > 0$ and Equation (12), are convex in the decision variables $\mathbf{Z}$ and $\mathbf{P}$. In order to ensure that the resulting control law is decentralized, we need to impose additional constraints. Toward this end, let $\mathcal{D} \subset \mathbb{R}^{M\times M}$ be the subset of diagonal positive definite matrices. Additionally, let $\mathcal{Z} = \{\mathbf{Y} \in \mathbb{R}^{N_{\mathcal{E}}\times M} : Y^{ij} = 0 \text{ if } i \neq j \text{ and } (i,j) \in \mathcal{V}\times\mathcal{V} - \mathcal{E}\}$. Then the additional constraints

$$\mathbf{Z} \in \mathcal{Z}, \quad \mathbf{P} \in \mathcal{D}, \tag{13}$$

which can be expressed as LMIs, can be imposed to achieve the desired decentralized structure in the controller. These two constraints are convex and can be expressed as LMIs. Hence, state-of-the-art LMI solvers can be used to construct control laws. Note that, for a general linear control system, the LMI (12) with the additional constraints (13) might not admit any solution $\mathbf{P}$, $\mathbf{Z}$. However, for the particular system (3), we have constructively established the existence of solutions in [10].

A necessary and sufficient condition for LMIs (12), (13) to be feasible is that the system $(\mathbf{A},\mathbf{B})$ (Equation (10)) is stabilizable. This system is not stabilizable on $\mathbb{R}^M$, since as noted earlier, the uncontrollable eigenvalue is at zero and the set $\mathcal{P}(\mathcal{V})$ is invariant for system (3). However, we require stability only on $\mathcal{P}(\mathcal{V})$. To deal with the lack of stabilizability on $\mathbb{R}^M$, one possibility is to use a model reduction approach by designing a lower-dimensional system that is controllable. This approach would be difficult, however, since it would not be possible to impose the structural constraints (13) on the controller matrix $\mathbf{K}$ in the original system, only on the controller matrix that is designed for the reduced-order system. Hence, we develop a simple alternative way to construct decentralized controllers, in which we artificially place the uncontrollable eigenvalue of the linear system in the open left half of the complex plane. To see this explicitly, let $(\mathbf{A},\mathbf{B})$ be a partially controllable system. Then there exists a nonsingular matrix $\mathbf{T} \in \mathbb{R}^{M\times M}$ such that

$$\tilde{\mathbf{A}} = \mathbf{T}\mathbf{A}\mathbf{T}^{-1} = \begin{bmatrix} \tilde{\mathbf{A}}_{11} & \tilde{\mathbf{A}}_{12} \\ \mathbf{0} & \tilde{\mathbf{A}}_{22} \end{bmatrix}, \ \tilde{\mathbf{B}} = \mathbf{T}\mathbf{B} = \begin{bmatrix} \tilde{\mathbf{B}} \\ \mathbf{0} \end{bmatrix}. \tag{14}$$

In our case, $\tilde{\mathbf{A}} = \mathbf{A} = \mathbf{0}$. However, this transformation is needed to convert $\tilde{\mathbf{B}}$ into the desired form. In order to design a controller for the system $(\mathbf{A},\mathbf{B})$, we can instead design a controller for another artificial system,

$$\tilde{\mathbf{A}}_\epsilon = \mathbf{T}\mathbf{A}\mathbf{T}^{-1} = \begin{bmatrix} \tilde{\mathbf{A}}_{11} & \tilde{\mathbf{A}}_{12} \\ \mathbf{0} & -\epsilon\mathbf{I} \end{bmatrix}, \ \tilde{\mathbf{B}} = \mathbf{T}\mathbf{B} = \begin{bmatrix} \tilde{\mathbf{B}} \\ \mathbf{0} \end{bmatrix} \tag{15}$$

for some $\epsilon > 0$, where $\mathbf{I}$ is the identity matrix of appropriate dimension. Note that the new artificial system, $(\tilde{\mathbf{A}}_\epsilon, \tilde{\mathbf{B}})$, has the same controllable eigenvalues as the original system, $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$, and all its uncontrollable eigenvalues are stable. We can perform the inverse transformation to represent the artificial system in the original coordinates:

$$\mathbf{A}_\epsilon = \mathbf{T}^{-1}\tilde{\mathbf{A}}_\epsilon\mathbf{T}. \tag{16}$$

Now let $\mathbf{F}$ be the gain matrix of a feedback control law that stabilizes the system $(\mathbf{A}_\epsilon, \mathbf{B})$. Then the following relation is satisfied, where $\sigma(\mathbf{M})$ is the spectrum of matrix $\mathbf{M}$:

$$\sigma(\mathbf{A}+\mathbf{BF})\backslash\sigma(\tilde{\mathbf{A}}_{22}) = \sigma(\mathbf{A}_\epsilon+\mathbf{BF})\backslash\{\epsilon\}.$$

This relation is advantageous from a computational point of view, since one can now directly impose the constraint of decentralized structure on the gain matrix $\mathbf{F}$ by designing the controller for the stabilizable artificial system $(\mathbf{A}_\epsilon, \mathbf{B})$ and then implementing it on the original system.

Combining the LMIs (12), (13), we obtain the following system of LMIs that need to be tested for feasibility:

$$\begin{gathered}\mathbf{P}\in\mathcal{D},\quad \mathbf{Z}\in\mathcal{Z},\quad \mathbf{P}>0,\\ \mathbf{PA}_\epsilon+\mathbf{BZ}+\mathbf{PA}_\epsilon^T+\mathbf{Z}^T\mathbf{B}^T<0.\end{gathered} \tag{17}$$

The solution to the above LMIs yields the stabilizing feedback law $\delta\mathbf{u}=\mathbf{F}\delta\mathbf{x}$, where $\mathbf{F}=\mathbf{ZP}^{-1}$.

It is straightforward to extend this approach to synthesize linear controllers with additional design specifications. For example, if it is necessary to impose limits on the norms of input signals or error signals, then constraints on system norms such as H_∞ and H_2 norms can be applied using the techniques in [9] for the design of linear controllers using LMIs. Additionally, controllers can be designed to achieve a desired rate of convergence to the equilibrium $\mathbf{x}^{eq}$. For example, suppose that we want the system $(\mathbf{A}_\epsilon, \mathbf{B})$ with feedback gain matrix $\mathbf{F}=\mathbf{ZP}^{-1}$ to exhibit a minimum decay rate α, a minimum damping ratio $\cos\theta$, and a maximum undamped natural frequency $r\sin\theta$. We can design such a controller using the concept of *D-stability control* [8, p. 108]:

Theorem III.3. *Suppose that there exist matrices* $\mathbf{P}>0$ *and* $\mathbf{Z}$ *such that*

$$\begin{gathered}\mathbf{M}=\mathbf{A}_\epsilon\mathbf{P}+\mathbf{BZ},\quad \mathbf{N}=(\mathbf{A}_\epsilon\mathbf{P}+\mathbf{BZ})^T,\\ \begin{bmatrix}-r\mathbf{P} & \mathbf{M}\\ \mathbf{N} & -r\mathbf{P}\end{bmatrix}<0,\\ \mathbf{M}+\mathbf{N}+2\alpha\mathbf{P}<0,\\ \begin{bmatrix}(\mathbf{M}+\mathbf{N})\sin\theta & (\mathbf{M}-\mathbf{N})\cos\theta\\ (\mathbf{N}-\mathbf{M})\cos\theta & (\mathbf{M}+\mathbf{N})\sin\theta\end{bmatrix}<0,\end{gathered} \tag{18}$$

Then if $\mathbf{F}=\mathbf{ZP}^{-1}$*, the pole locations* $z\in\mathbb{C}$ *of the matrix* $\mathbf{A}_\epsilon+\mathbf{BF}$ *satisfy* $|z|\le r$ *and Re* $z\le-\alpha$*.*

B. Nonlinear Polynomial Controller

In [10], we showed that the set of decentralized nonlinear control laws that solve Problem II.1 is nonempty by explicitly constructing one such control law. To prove its stability, we demonstrated that the following function is a Lyapunov function for system (5):

$$V(\mathbf{x})=\frac{1}{2}\left(\mathbf{x}^T\mathbf{Dx}-(\mathbf{x}^{eq})^T\mathbf{Dx}^{eq}\right), \tag{19}$$

where $\mathbf{D}=diag([1/x_1^{eq}\ \ 1/x_2^{eq}\ \ ...\ \ 1/x_M^{eq}])$. This Lyapunov function is commonly used in multi-agent consensus protocols [15]. Here, we present an algorithmic procedure for constructing other nonlinear control laws that solve Problem II.1, using the function (19) in the construction. This procedure allows us to introduce additional constraints to improve the performance of the closed-loop system, as in the linear controller design procedure. We will construct control laws that are polynomial functions of the system state. This allows us to frame Problem II.1 as a polynomial optimization problem that can be solved using SOSTOOLS [19], a MATLAB toolbox for solving sum-of-squares (SOS) programs. SOSTOOLS is widely used to provide algorithmic solutions to problems with polynomial non-negativity constraints that are otherwise difficult to solve. The non-negativity constraints are relaxed to a test for the existence of an SOS decomposition, and this test is performed using semidefinite programming. We note that our procedure is just one possible method for constructing the control laws.

We first pose Problem II.1 as an optimization problem.

Problem III.4. *Let* $\mathbb{R}[\mathbf{x}]$ *denote the set of polynomials (not necessarily positive), and let* Σ_s *denote the set of SOS polynomials. Given system* (5) *with the matrix* $\mathbf{B}_e$ *defined as in* (4)*, and given the function* $V(\mathbf{x})$ *in Equation* (19)*, find* $k_e(\mathbf{x})\in\mathbb{R}[\mathbf{x}]$ *such that*

$$k_e(\mathbf{x})\ge 0, \tag{20}$$

$$k(\mathbf{x}^{eq})=0, \tag{21}$$

$$\nabla V(\mathbf{x})^T F(\mathbf{x})\le 0 \tag{22}$$

for all $\mathbf{x}\in\mathcal{P}(\mathcal{V})$*, where* $F(\mathbf{x})=\sum_{e\in\mathcal{E}}k_e(\mathbf{x})\mathbf{B}_e\mathbf{x}$*.*

In Problem III.4, the candidate Lyapunov function $V(\mathbf{x})$ is fixed, and an appropriate control law is constructed such that $V(\mathbf{x})$ is indeed a Lyapunov function for the closed-loop system (5). We can easily confirm that $V(\mathbf{x})=0$ at the equilibrium $\mathbf{x}^{eq}$, and simple algebraic manipulation shows that $V(\mathbf{x})>0$ for all $\mathbf{x}\in\mathcal{P}(\mathcal{V})\backslash\{\mathbf{x}^{eq}\}$. However, establishing the local negative definiteness of the gradient of $V(\mathbf{x})$ on the simplex $\mathcal{P}(\mathcal{V})$, as specified by the inequality (22), is a relatively difficult constraint to encode in SOSTOOLS. We enforce this constraint by using the theory of *positivestellansatz* [6]: *Given a function* $f(\mathbf{z})$*, where* $\mathbf{z}\in\mathbb{R}^n$*, and a set* $S\subset\mathbb{R}^n$*, is* $f(\mathbf{z})\ge 0$ *(or, alternatively, is* $f(\mathbf{z})<0$*) for every* $\mathbf{z}\in S$*?* In this formulation, S is a *semialgebraic set*, which is defined as:

$$S:=\{\mathbf{z}\in\mathbb{R}^n\mid p_i(\mathbf{z})\ge 0,\ q_j(\mathbf{z})=0,\ i,j\in\mathbb{N}\}, \tag{23}$$

where p_i and q_i are polynomial functions of the state variable $\mathbf{z}$ and $\mathbb{N}$ is the set of natural numbers. S may also include constraints of the form $p_i(\mathbf{z})<0$ and $q_j(\mathbf{z})\ne 0$.

Schmudgen's positivestellansatz [20], stated below, gives sufficient conditions for the positivity of $f(\mathbf{z})$ on a compact semialgebraic set $S\subset\mathbb{R}^n$.

Theorem III.5. *Suppose that the semialgebraic set* (23) *is compact. If* $f(\mathbf{z})\ge 0$ *for all* $\mathbf{z}\in S$*, then there exist* $t_j\in\mathbb{R}[\mathbf{z}]$ *and* $s_0, s_i, s_{ij}, s_{ijk}, ...\in\Sigma_s$ *such that*

$$\begin{aligned}f=\ &\sum_j t_jq_j+s_0+\sum_i s_ip_i+\sum_{i,j:i\ne j}s_{ij}p_ip_j+\\ &\sum_{i,j,k:i\ne j\ne k}s_{ijk}p_ip_jp_k+...\end{aligned} \tag{24}$$

In our case, the simplex $\mathcal{P}(\mathcal{V})$ is a compact semialgebraic set in $\mathbb{R}^M$ of the form (23), in which the inequalities $p_i\ge$

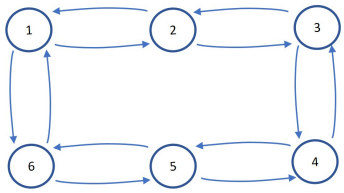


Fig. 1: Six-vertex bidirected graph.

0, $i = 1, ..., M$, are given by $x_1 \geq 0$, ... , $x_M \geq 0$, and the equality $q_1 = 0$ is given by $1 - x_1 - ... - x_M = 0$. Thus, according to Theorem III.5, verifying the inequality (22) for all $\mathbf{x} \in \mathcal{P}(\mathcal{V})$ reduces to searching for $t_1 \in \mathbb{R}[\mathbf{x}]$ and $s_0, s_i, s_{ij}, s_{ijk}, ... \in \Sigma_s$ such that

$$-\nabla V(\mathbf{x})^T F(\mathbf{x}) = t_1 q_1 + s_0 + \sum_i s_i p_i + \sum_{i,j:i\neq j} s_{ij} p_i p_j + ... \tag{25}$$

We can use SOSTOOLS to find polynomials t_1 and $s_0, s_i, s_{ij}, ...$ that satisfy Equation (25).

We note that Problem III.4 could alternatively be formulated to search for both the Lyapunov function and the control law simultaneously. Since this would render the optimization problem bilinear in the variables $V(\mathbf{x})$ and $k_e(\mathbf{x})$, it would be possible to solve the problem by iterating between these variables. However, this approach does not guarantee convergence to a solution.

IV. NUMERICAL SIMULATIONS

We computed linear and nonlinear feedback controllers for the closed-loop system (5) to redistribute populations of $N = 20$ and $N = 1200$ agents on the six-vertex bidirected graph in Fig. 1. The linear controller was computed using the method described in Section III-A with the LMIs (17), (18). In the LMIs (18), we set $\alpha = 1$, $\cos(\theta) = 0.707$, and $r = 1.5$ for desired transient response characteristics. The nonlinear controller was constructed by using SOSTOOLS to solve Problem III.4, as described in Section III-B. For both controllers, the initial distribution was $\mathbf{x}^0 = [0.2\ \ 0.1\ \ 0.2\ \ 0.15\ \ 0.2\ \ 0.15]^T$, and the desired distribution was $\mathbf{x}^{eq} = [0.1\ \ 0.2\ \ 0.05\ \ 0.25\ \ 0.15\ \ 0.25]^T$.

The solution of the mean-field model (5) with each of the two controllers and the trajectories of a corresponding stochastic simulation are compared in Fig. 2. For ease of comparison, the total agent populations were normalized to 1 in both the mean-field model and the stochastic simulation. We observe from the plots that both controllers drive the agent distributions to the desired equilibrium distribution, with the nonlinear controller yielding much slower convergence to this equilibrium than the linear controller. This is because the inequalities in problem III.4 only guarantee asymptotic stability of the system (5). We note that if faster convergence is desired, then this could be encoded as constraint in SOSTOOLS.

The underlying assumption of using the mean-field model (5) is that the swarm behaves like a continuum. That is, the ODE (5) is valid as the number of agents $N \to \infty$ [13]. Hence, it is imperative to check the performance of the feedback controller for different agent populations. We observe that the stochastic simulation follows the ODE solution closely in all four simulations, and that the stochastic simulation exhibits smaller fluctuations about the ODE solution when the agent population is increased from $N = 20$ to $N = 1200$. In addition, in all simulations, the numbers of agents in each state remain constant after some time; in the case of 20 agents, the fluctuations stop earlier than in the case of 1200 agents.

The linear controllers are computationally less expensive to construct than the nonlinear ones, and hence they can be more easily scaled with the number of states and control inputs. To illustrate this scalability, we computed a linear controller to redistribute $N = 5000$ agents on a 65-vertex graph with a two-dimensional grid structure to spell out the letters *ACS*, and we ran a stochastic simulation of the resulting control system. Fig. 3a shows the initial distribution of the agents on a two-dimensional domain, in which each partitioned region corresponds to a vertex in the graph. Initially, 1800 agents are in a single state (the bottom left region), and the rest are distributed equally among the other 64 states. We assume that agents switch between regions instantaneously once they decide to execute a transition. Fig. 3b shows that at time $t = 1000$ s, the agent distribution closely matches the desired equilibrium distribution. A movie of the agent redistribution is shown in the *Video attachment*.

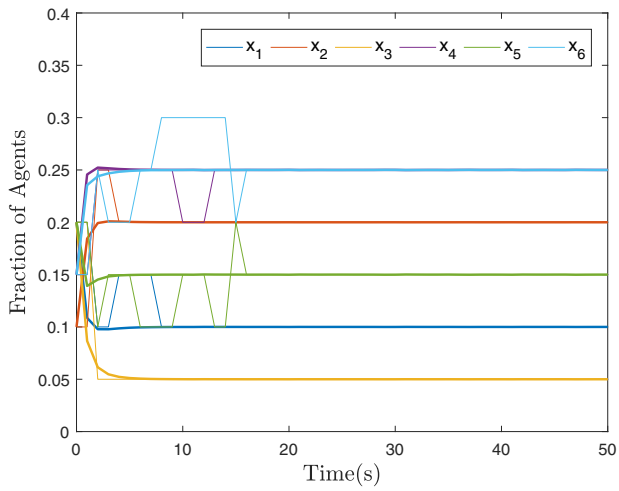
V. MULTI-ROBOT EXPERIMENTAL RESULTS

A. Experimental Testbed

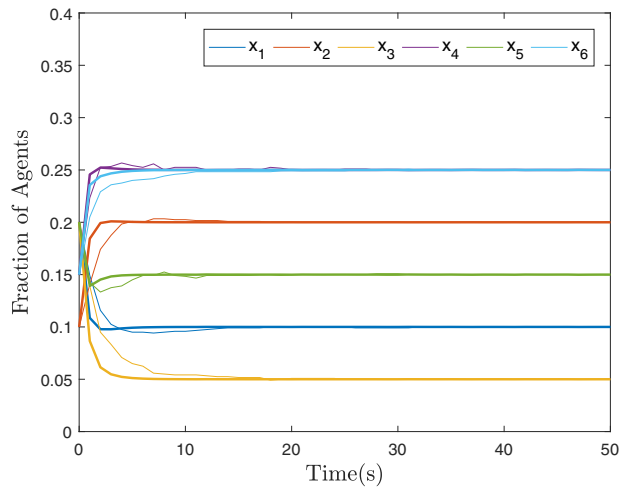
We evaluated our linear and nonlinear controllers in a scenario in which a group of small differential-drive robots must reallocate themselves among four regions. For these experiments, we used ten Pheeno mobile robots [21], each equipped with a Raspberry Pi 3 computer, a Teensy 3.1 microcontroller board, a Raspberry Pi camera, six IR sensors around its perimeter, and a bottom-facing IR sensor. Pheeno is compatible with ROS, the Robot Operating System, which facilities the implementation of advanced algorithms with our multi-agent system.

For the experiments, we used a 2 m $\times$ 2 m confined arena, shown in Fig. 4, that was divided into four regions of equal size. These regions were labeled 1, 2, 3, and 4 according to the green numbers in Fig. 5. Each region corresponds to a vertex of a bidirected graph that defines the robots' possible transitions between the regions. Robots can move between adjacent regions but not diagonally; i.e., there are no edges between vertices 1 and 4 and between vertices 2 and 3. The walls along the borders of regions 1, 2, 3, and 4 are colored purple, cyan, green, and red, respectively, as shown in Fig. 4. In addition, regions 1 and 4 have a black floor, and regions 2 and 3 have a white floor. These color features were included to enable each robot to identify its current state (region) through image processing and IR measurements.

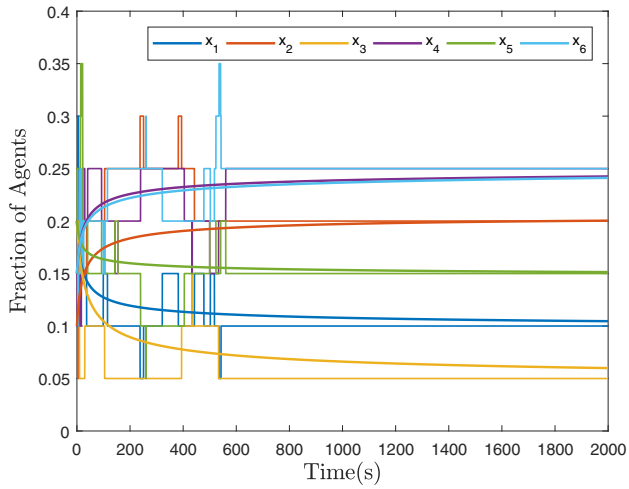
A Microsoft LifeCam camera was mounted over the testbed to obtain overhead images of the experiments. The robots were marked with identical yellow square tags, which were detected using the camera (see the red circles in Fig. 5). A central computer processed images obtained by the camera and communicated with all the robots over WiFi. The robots



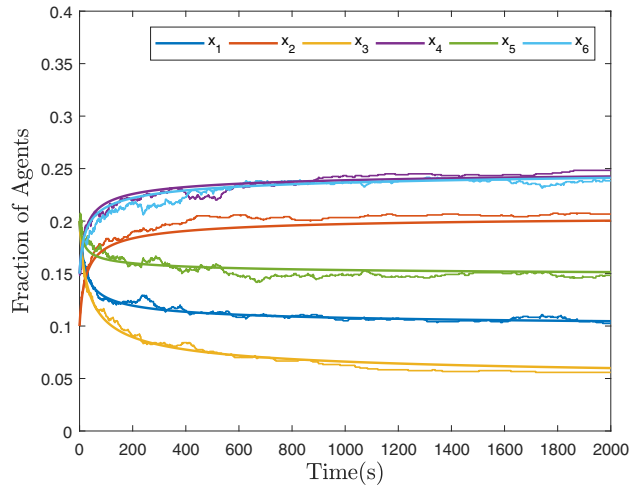
(a) Linear controller with $N = 20$ agents



(b) Linear controller with $N = 1200$ agents

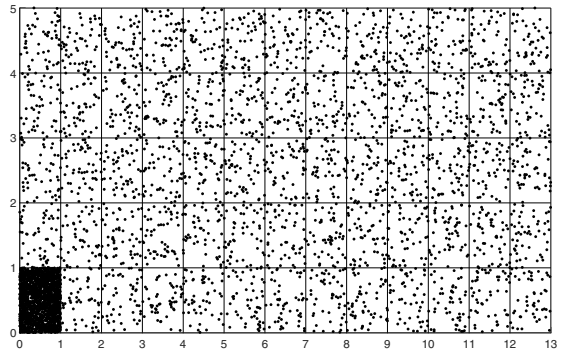


(c) Nonlinear controller with $N = 20$ agents

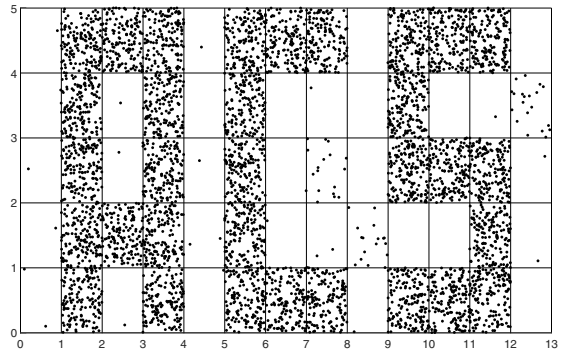


(d) Nonlinear controller with $N = 1200$ agents

Fig. 2: Trajectories of the mean-field model *(thick lines)* and the corresponding stochastic simulations *(thin lines)*.



(a) Distribution at $t = 0$ s



(b) Distribution at $t = 1000$ s

Fig. 3: Snapshots of $N = 5000$ agents redistributing over a 65-vertex graph during a stochastic simulation of the closed-loop system with a linear controller.

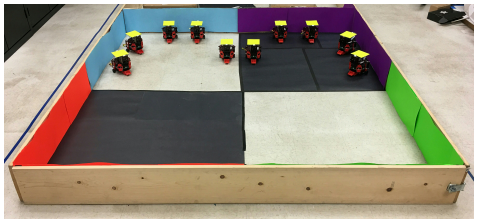


Fig. 4: Multi-robot experimental testbed.

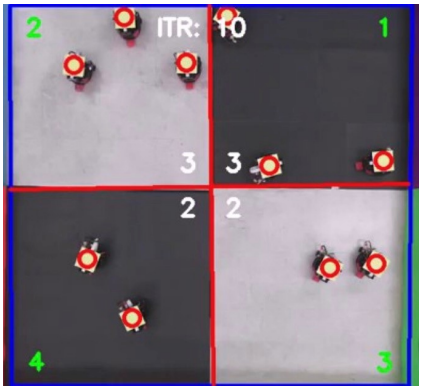


Fig. 5: Overhead camera view of the testbed during an experimental trial.

did not have access to information about their positions in a global frame.

B. ROS Setup

The entire setup utilized ROS middleware. The central computer runs the ROS Master and two ROS nodes: an *overhead camera node* to process images of the testbed from the overhead camera, and a *transition control node* to initiate or end an iteration. The *overhead camera node* calculates the number of robots in each region. This calculation does not require the identification of individual robots; instead, the node uses color detection to count the number of yellow identification tags inside each region on the testbed. These numbers are then converted into robot densities in each state and are published on a ROS topic. The *transition control node* monitors the state transition iterations. The node ends an iteration when every robot has reached its desired state.

Each robot runs three ROS nodes: a *sensor node*, a *camera node*, and a *controller node*. The *sensor node* publishes data from all the robot's IR sensors on their corresponding topics and drives the robot's motors by subscribing to movement command topics. The *camera node* publishes raw images from the robot's onboard cameras. Finally, the *controller node* runs the motion control scheme for the robot, described in Section V-C. This node receives the state densities from the central computer's *overhead camera node*. The robot computes the next desired state using the controller input and has to decide whether to stay in its current region or transition to another one.

C. Robot Motion Controller

Each robot starts in one of the four regions (states), according to the specified initial condition, and is programmed with the desired equilibrium distribution $\mathbf{x}^{eq}$ and the set of transition rates $k_e(\mathbf{x})$ that have been designed by the central computer using one of the procedures described in Section III. The robots know their initial region and update their region after each iteration. At the start of each iteration, the robots receive state feedback $\mathbf{x}$, the current robot densities in each region, from the *overhead camera node.* Using this information, each robot computes its probability $k_e(\mathbf{x})\delta t$, where $\delta t = 0.1$, of transitioning to an adjacent spatial region $T(e)$ within the next iteration. This stochastic decision policy is executed by the robot using a random number generator. If a robot decides to transition to another region, it searches for the color on the wall of that region. As soon as it finds the color, it moves ahead along a straight path. If an encountered object obstructs its path, the robot avoids it and reorients itself toward the target region. Since the robot's onboard camera is unable to detect changes in depth, we assigned each region to have a white or black floor, which can be identified by a bottom-facing IR sensor on each robot. The robot detects that it has entered the target region when it identifies a change in the floor color. The robot then moves forward a small distance, which prevents robots from clustering on the region boundaries, and stops moving. Finally, the robot sends a *True* signal to the *transition control node*. This node initiates the next iteration once it receives a *True* signal from all the robots. The entire process is repeated until the desired distribution is reached by the robots.

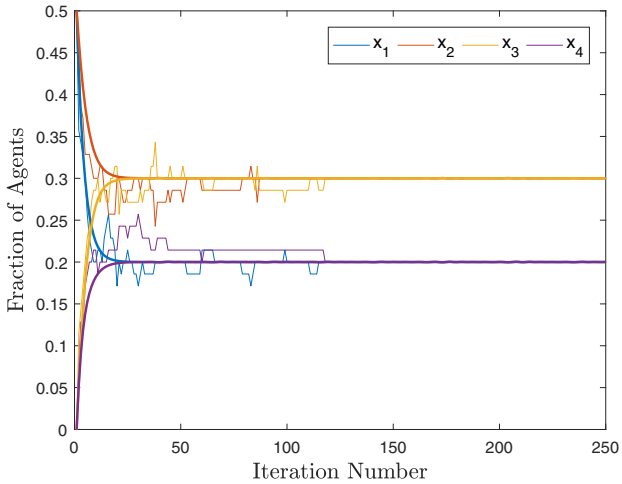
D. Results

We computed linear and nonlinear feedback controllers for the experiments in the same way that we computed the controllers in Section IV for the simulations. The controllers were designed to redistribute a population of $N = 10$ robots on the four-vertex bidirected graph corresponding to the arena. The initial distribution was defined as $\mathbf{x}^0 = [0.5\ \ 0.5\ \ 0\ \ 0]^T$. For the linear controller, the desired distribution was $\mathbf{x}^{eq} = [0.2\ \ 0.3\ \ 0.3\ \ 0.2]^T$, and for the nonlinear controller, it was $\mathbf{x}^{eq} = [0.3\ \ 0.2\ \ 0.2\ \ 0.3]^T$.

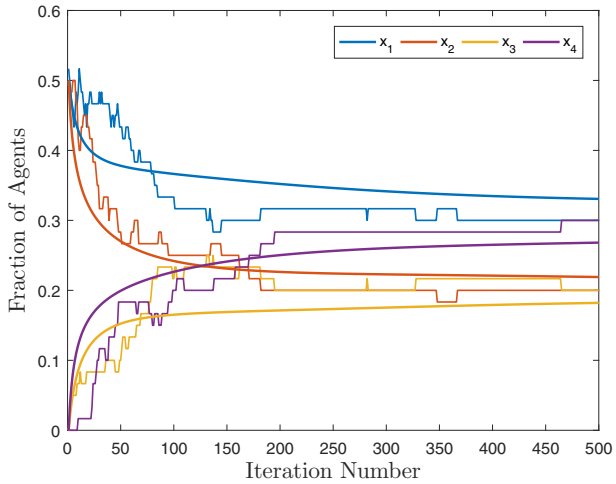
Fig. 6 shows the solution of the mean-field model (5) with each of the two controllers and the corresponding robot populations in each state from the experiments, averaged over five trials. For ease of comparison, the total robot populations were normalized to 1. The plots show that the robots successfully redistribute themselves to the target distribution, as predicted by the mean-field model. As in the numerical simulations in Fig. 2, the nonlinear controller produces a slower convergence rate to equilibrium than the linear controller. Movies of the robot experiments with both the linear and nonlinear controllers are shown in the *Video attachment*.

VI. CONCLUSION

In this paper, we have presented computational procedures for constructing decentralized state feedback controllers which stabilize a swarm of agents to a target distribution among a



(a) Linear controller with $N = 10$ robots



(b) Nonlinear controller with $N = 10$ robots

Fig. 6: Trajectories of the mean-field model *(thick lines)* and the robot population fraction in each state, averaged over five experimental trials *(thin lines)*.

set of states. The agents switch stochastically between states according to a continuous-time Markov chain. We designed linear controllers using LMI-based techniques and nonlinear controllers using results from polynomial optimization. We validated these controllers through numerical simulations with different numbers of agents and graph sizes and through physical experiments with ten robots. In future work, we will investigate the design of linear feedback controllers that are globally asymptotically stabilizing. We will also design nonlinear feedback controllers that are optimized for a fast convergence rate to the desired equilibrium.

References

[1] Behcet Acikmese and David S Bayard. A Markov chain approach to probabilistic swarm guidance. In *American Control Conference (ACC)*, pages 6300–6307. IEEE, 2012.

[2] William Agassounon, Alcherio Martinoli, and Kjerstin Easton. Macroscopic modeling of aggregation experiments using embodied agents in teams of constant and time-varying sizes. *Autonomous Robots*, 17(2):163–192, 2004.

[3] Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y Hadaegh. Inhomogeneous markov chain approach to probabilistic swarm guidance algorithm. In *5th Int. Conf. Spacecraft Formation Flying Missions and Technologies,(Munich, Germany)*, 2013.

[4] Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y Hadaegh. Probabilistic and distributed control of a large-scale swarm of autonomous agents. *IEEE Transactions on Robotics*, 2017.

[5] Spring Berman, Ádám Halász, M Ani Hsieh, and Vijay Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, 25(4):927–937, 2009.

[6] Graziano Chesi. *Domain of attraction: analysis and control via SOS programming*, volume 415. Springer Science & Business Media, 2011.

[7] Nazlı Demir, Utku Eren, and Behçet Açıkmeşe. Decentralized probabilistic density control of autonomous swarms with safety constraints. *Autonomous Robots*, 39(4):537–554, 2015.

[8] Guang-Ren Duan and Hai-Hua Yu. *LMIs in control systems: analysis, design and applications*. CRC press, 2013.

[9] Geir E Dullerud and Fernando Paganini. *A course in robust control theory: a convex approach*, volume 36. Springer Science & Business Media, 2013.

[10] Karthik Elamvazhuthi, Shiba Biswal, Vaibhav Deshmukh, Kawski Matthias, and Spring Berman. Mean field controllability and decentralized stabilization of Markov chains (Accepted). In *Decision and Control (CDC), 2017 IEEE 56th Conference on*. IEEE, 2017.

[11] M Ani Hsieh, Ádám Halász, Spring Berman, and Vijay Kumar. Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence*, 2(2):121–141, 2008.

[12] Peter Kingston and Magnus Egerstedt. Index-free multi-agent systems: An eulerian approach. *IFAC Proceedings Volumes*, 43(19):215–220, 2010.

[13] Vassili N Kolokoltsov. *Nonlinear Markov processes and kinetic equations*, volume 182. Cambridge University Press, 2010.

[14] Kristina Lerman, Chris Jones, Aram Galstyan, and Maja J Matarić. Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, 25(3):225–241, 2006.

[15] Frank L Lewis, Hongwei Zhang, Kristian Hengster-Movric, and Abhijit Das. *Cooperative control of multi-agent systems: optimal and adaptive design approaches*. Springer Science & Business Media, 2013.

[16] Alcherio Martinoli, Kjerstin Easton, and William Agassounon. Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *The International Journal of Robotics Research*, 23(4-5):415–436, 2004.

[17] T William Mather and M Ani Hsieh. Synthesis and analysis of distributed ensemble control strategies for allocation to multiple tasks. *Robotica*, 32(02):177–192, 2014.

[18] Dejan Milutinovic and Pedro Lima. Modeling and optimal centralized control of a large-size robotic population. *IEEE Transactions on Robotics*, 22(6):1280–1285, 2006.

[19] Stephen Prajna, Antonis Papachristodoulou, and Pablo A Parrilo. Introducing sostools: A general purpose sum of squares programming solver. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, pages 741–746. IEEE, 2002.

[20] Konrad Schmüdgen. The K-moment problem for compact semi-algebraic sets. *Mathematische Annalen*, 289(1):203–206, 1991.

[21] Sean Wilson, Ruben Gameros, Michael Sheely, Matthew Lin, Kathryn Dover, Robert Gevorkyan, Matt Haberland, Andrea Bertozzi, and Spring Berman. Pheeno, a versatile swarm robotic research and education platform. *IEEE Robotics and Automation Letters*, 1(2):884–891, 2016.