IRIDIA

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle

Global-to-Local Design for Self-Organized Task Allocation in Swarms

G. Valentini, H. Hamann, and M. Dorigo

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2016-002

March 2016

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2016-002

Global-to-Local Design for Self-Organized Task Allocation in Swarms

Gabriele Valentini[†], Heiko Hamann[‡] and Marco Dorigo[†]
[†]IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
{gvalenti, mdorigo}@ulb.ac.be
[‡]Heinz Nixdorf Institute, Dept. of Computer Science, University of Paderborn, Paderborn, Germany
heiko.hamann@uni-paderborn.de

ABSTRACT

We propose a global-to-local design methodology to compose heterogeneous swarms for self-organized task allocation. We consider embodied agents with only local knowledge and local communication driven by a simple reactive-control architecture (i.e., local level). We define finitely many agent controllers and focus on the macroscopic dynamics (i.e., global level) of behaviorally heterogeneous swarms. The user inputs the desired swarm allocations as the stationary distribution of agents over two tasks. Given the input, we mathematically derive the composition of a heterogeneous swarm that approximates the requirements of the user. We investigate our methodology over several case studies and validate our results with multi-agent simulations.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: intelligent agents, multiagent systems

General Terms

Algorithms, Design, Theory

Keywords

global-to-local design; self-organized task allocation; heterogeneous swarms; swarm robotics; swarm intelligence

1. INTRODUCTION

The primary challenge that hinders the spread of applications with large collections of embodied agents [13, 32] is the design of individual agent controllers that lead to a desired collective behavior. The canonical, local-to-global approach [6] includes a trial and error refinement of individual agent control rules followed by the macroscopic analysis of the resulting swarm behavior [26, 18] or by the formal verification of specific properties of interest [28, 5, 22, 23]. Designing agent controllers that generate a desired swarm behavior has proven challenging, and at present, only a few methods exist and these methods are tailored to specific scenarios (e.g., task allocation [3, 4, 11], formation control [10], self-assembly [29, 21, 31], collective construction [39]).

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), John Thangarajah, Karl Tuyls, Stacy Marsella, Catholijn Jonker (eds.), May 9–13, 2016, Singapore.*

To tackle this challenge, we propose a novel global-to-local design approach. Our key idea is to compose a heterogeneous swarm using groups of behaviorally different agents in a way that the resulting swarm behavior approximates a user input representing the desired behavior of the swarm. This idea is related to the concept of population coding from neurosciences where certain phenomena (e.g., direction of movement) result from the average of different individual contributions from a population of neurons [15]. Similarly to population coding [30], we derive a global-to-local design method that performs a function approximation of the user input by linear combination of basis vectors.

We illustrate this idea by defining a prescriptive design method for self-organized task allocation [24, 1, 25]. Specifically, we tackle variants of the *single-task robots, multi-robot tasks* problem (ST-MR) [16] with static, sequential, and periodic swarm allocations. In the ST-MR class of problems each agent can execute only one task at a time and each task requires several agents to be performed.

We consider a scenario where a user wishes to design a swarm that allocates its members over two alternative tasks. A particular swarm allocation corresponds to a partitioning of the agents in the swarm in two subgroups, one for each of the two tasks. The user provides a description of the desired swarm allocations as input. Specifically, the user input consists of a probability distribution over swarm allocations. In order to define a specif task allocation scenario, the user manipulates the number and positions of the modes of the distribution (i.e., local maxima) with each of them specifying a desired swarm allocation. The user might specify a static task allocation scenario by providing as input a unimodal distribution. A sequential task allocation scenario is defined instead using a sequence of unimodal distributions over swarm allocations and a criterion to decide when to move from the current distribution to the next in the sequence. Finally, a user can define a periodic task allocation scenario using a distribution with two or more modes; in this case, the swarm periodically changes the allocation of its members as specified by the modes of the user input.

Our global-to-local design method proceeds as follows. We define an arbitrarily large number of agent controllers, that is, sets of predefined control rules. For each agent controller, we derive a basis vector that models its global-level contribution to the swarm dynamics. The probability distribution over swarm allocations given by the user as input defines the desired asymptotic behavior of the swarm. Given this input, we compute a response vector describing the corresponding transient behavior. That is, the transient dynamics of a swarm that asymptotically converges to the user input. We use the response vector as a reference to select the necessary agent controllers. Finally, we systematically search for a proper composition of a heterogeneous swarm by estimating the coefficients in a lasso regression [36] between the response vector and a linear combination of basis vectors. We use penalized regression to select the agent controllers required in the swarm and the value of the coefficients to define their proportions.

2. RELATED WORKS

Starting from the idea of behavioral heterogeneity, in this paper we propose a prescriptive approach to design swarms for ST-MR task allocation [16] with both static and time-variant swarm allocations. This problem is generally known as the coalition formation problem and has been extensively studied in the multi-agent community [33, 12]. Standard multi-agent approaches, however, require complex cooperation strategies with a priori negotiation or bidding for task assignments and unconstrained communication. These approaches are not suitable for large-scale swarms of unreliable agents.

Conversely, the swarm intelligence community developed a rich framework of algorithms for self-organized task allocation that are suitable for unreliable, embodied agents. Instead of forming a priori coalitions, self-organized multi-agent systems achieve task allocation as a result of the continuous interaction among agents and between agents and the environment. Popular approaches include threshold-based algorithms [1, 27] and recruitment strategies inspired by the foraging behavior of ant colonies [24, 25]. More recently, Castillo-Cagigal and colleagues [9] investigated periodic binary task allocation and proposed a self-organized synchronization strategy whose macroscopic behavior results in a bimodal distribution of robots over two tasks—a scenario that we also consider in this paper. All these studies, however, employ local-to-global design approaches. Although the proposed strategies are often supported by descriptive macroscopic models [2], these studies do not provide a prescriptive design method as the one proposed in this paper.

A notable exception is the method proposed by Berman et al. [3] for the design of task allocation. In their work, the authors consider problems with more than two tasks. Each agent decides to switch tasks independently of other agents in the swarm and therefore agents do not interact with each other. Their method, based on a linear continuous model, optimizes a set of transition rates between pairs of tasks that define a unique agent controller with the aim to converge as quickly as possible to a given swarm allocation. Differently from the probability distribution over swarm allocations considered here, the authors assume as input a single swarm allocation corresponding to the mode of our unimodal scenario (see Section 5.1). Due to the lack of interactions among agents, their method cannot achieve the non-linear, oscillatory dynamics of the swarm that we describe in our multimodal scenarios (see Section 5.2). The approach of Berman et al. has also been extended to incorporate feedback gathered from the environment by individual agents [4, 11]. Agents keep track of the number of successfully completed tasks and report this information to a centralized authority (called the hive) that, in turn, updates the parameters of their stochastic control policy. Differently, our design approach provides a completely self-organized solution that does not require any centralized authority.

Most existing global-to-local design approaches focus on self-assembly [29] and formation control problems [10]. The method proposed by Klavins makes use of graph grammars and optimizes their execution rates to design a system that self-assembles into simple controlled shapes [21]. More recently, Rubenstein et al. [31] proposed a distributed algorithm for self-assembly and experimented with a swarm of more than a thousand robots. In their study, robots are given a blueprint of the desired shape and follow only local cues to incrementally position themselves according to the blueprint. In collective construction, Werfel and colleagues [39] propose a compilation method that decomposes a user-specified structure into a set of construction paths that robots follow to build a desired artifact. Similarly to us, all these studies use optimization methods to explore a (possibly constrained) design space. However, these methods are tailored for their respective applications and substantially differ from the idea of composing behavioral heterogeneity. The potential of behavioral heterogeneity has been recently investigated in an aggregation scenario [20]. In this study, the authors show, by means of evolutionary computation techniques, that heterogeneous swarms can outperform their homogeneous counterparts. Finally, our method has some similarities with the approach used by Hamann et al. [17] to analyze collective motion in locust swarms; the authors use a linear combination of polynomials to fit a network model to macroscopic measurements of simulations. In their approach, the regression coefficients provide information about the spatial distribution of agents in the swarm.

3. AGENT CONTROLLERS

We consider the problem of designing a swarm of N agents that allocates its members to a pair of tasks (henceforth, referred to as task 0 and task 1) as defined by a user input. Let $(X, N-X)$ represent a swarm allocation, where $X \in \mathcal{X}$ is the number of agents allocated to task 0 (respectively, $N-X$ to task 1), and $\mathcal{X} = \{0, 1, \dots, N\}$ is the system state space. The user inputs a desired stationary probability distribution $\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \dots, \boldsymbol{\pi}_N)$, $\boldsymbol{\pi}_i > 0$, of agents over the two tasks. Entries $\boldsymbol{\pi}_i$, $i \in \mathcal{X}$, give the probability that the swarm allocation is $(i, N-i)$, and each mode of $\boldsymbol{\pi}$ (i.e., local maxima) defines a desired swarm allocation $(X, N-X)$. The number of modes of the user input determines the particular variant of the task allocation scenario. A distribution $\boldsymbol{\pi}$ with one unique mode corresponds to a single swarm allocation; given a sequence $\boldsymbol{\pi}^1, \boldsymbol{\pi}^2, \dots$ of such distributions, we can design swarms for sequential task allocation. When the user input $\boldsymbol{\pi}$ is a multimodal distribution, the behavior required by the user corresponds to periodic task allocation. With a certain frequency, the swarm changes the allocation of its members according to the different swarm allocations determined by the modes of $\boldsymbol{\pi}$.

We consider agents with only local perception of their environment and local agent-to-agent communication. By building on these limited capabilities, we define a recipe to enumerate finitely many different agent controllers. We achieve this by considering a template of a control rule that can be instantiated with different configurations and that allows us to enumerate different agent controllers. To do so, we abstract from the domain-specific actions necessary to an agent to execute a particular task. Instead, we focus on the agent interactions and the decision-making necessary to fulfill the swarm allocations desired by the user. We consider a system where tasks are uniformly distributed in a closed environment in which agents *i)* move randomly while working on either of the two tasks, *ii)* have the ability to perceive the allocation of their neighbors, and *iii)* to trigger a change in their allocation (one at a time).

Agents act stochastically and asynchronously. They repeatedly apply two control rules: *self-switching* and *switch-or-recruit*. When executing the self-switching rule, an agent changes its current allocation to the alternative task. The global effect of the self-switching rule can be assimilated to the spontaneous switching behavior of unreliable agents subject to internal noise [14, 40]. Using the switch-or-recruit rule, an agent has a greater influence on the current swarm allocation. It can decide to either increase or decrease the number of agents allocated to a task by one unit. As a function of its current allocation and those of its neighbors, the agent either self-switches to the other task or recruits a neighbor from those with the alternative allocation. When the agent acts as a recruiter, the recruited neighbor always switches its task allocation and it does so independently of its internal state and of its actual agent controller. That is, passively recruited agents always switch their task allocation without objections. Control rules are executed randomly by individual agents: self-switching with rate σ and switch-or-recruit with rate ρ (respectively, with probabilities $p_\sigma = \sigma/(\sigma+\rho)$ and $p_\rho = \rho/(\sigma+\rho)$).

Agent controllers $\langle G; b\rangle$, $b \in \{1, \ldots, 2^{G-1}\}$, differ from each other by the logical function $\Delta_{G,b}$ used to determine the global effect of the switch-or-recruit rule. Function $\Delta_{G,b}$ takes as input a group of task allocations of size $G \ll N$; this group includes the task allocation of the agent applying the switch-or-recruit rule and that of its $G-1$ neighbor agents. Parameter b is an index that ranges over all possible agent controllers based on the same group size G. For a group of task allocations of size G, we have $G+1$ possible group compositions. We do not assign any action to homogeneous groups (i.e., groups with either 0 or G agents allocated to task 0). Therefore, we have that $\Delta_{G,b}$ has $G+1$ possible inputs and 3 possible outputs (i.e., switch allocation, recruit a neighbor, no action). Moreover, since the no-action is fixed in each agent controller, we obtain 2^{G-1} possible functions $\Delta_{G,b}$. Function $\Delta_{G,b}$ is defined as $\Delta_{G,b} = (\Delta_1, \ldots, \Delta_{G-1})$, $\Delta_i = \pm 1$. Δ_i gives the change of agents allocated to task 0 when an agent applies the switch-or-recruit rule over a group of allocations that contains $i \in \{1, \ldots, G-1\}$ entries for task 0. Given a particular choice of values for parameters G and b, we set $\Delta_i = +1$ if the i-th bit of b (expressed in the binary numeral system) equals 0; otherwise, we set $\Delta_i = -1$. Table 1 shows an example of an agent controller defined by $\Delta_{3,1} = (+1, -1)$; in this case, the switch-or-recruit rule corresponds to the majority rule often used by robot swarms [38]. By enumerating all $\Delta_{G,b}$ for increasing values of G we obtain an arbitrary large set $\mathcal{B} = \{\langle G_1; 1\rangle, \ldots, \langle G_1; 2^{G_1-1}\rangle, \ldots \langle G_i; 1\rangle, \ldots, \langle G_i; 2^{G_i-1}\rangle, \ldots\}$ of different agent controllers.

It is relevant to note three properties that result from the above defined controllers. Firstly, agent controllers are independent of the controllers of neighboring agents and only require to know their task allocations. Secondly, the interplay between self-switching and direct recruitment eases the mixing of task allocations among agents with different controllers. Thirdly, due to self-switching, the resulting decision-making process is ergodic which prevents its absorption at extreme states where all agents are allocated to one of the two tasks [37]. These properties are fundamental for the definition of our global-to-local design method because they allow us to correctly predict the global behavior of a heterogeneous swarm.

a	$\mathcal{N}_0$	$\Delta_{G,b}$	action	a	$\mathcal{N}_0$	$\Delta_{G,b}$	action
0	0	Δ_1	switch	1	0	–	–
0	1	Δ_2	recruit	1	1	Δ_1	recruit
0	2	–	–	1	2	Δ_2	switch

Table 1: Example of logical function $\Delta_{G,b}$ with $G=3$ and $b=1$. Symbol a gives the current task allocation of the focal agent, $\mathcal{N}_0$ is the number of neighbors allocated to task 0, and ($\Delta_1 = +1, \Delta_2 = -1$) define the outcome of $\Delta_{G,b}$ (in this case a majority rule). Symbol '–' represents no action.

Multi-agent simulation.

We have developed a simple microscopic multi-agent simulator to validate our design method. In our simulations, agents consist of situated mass-less points moving within an environment of 100×100 space units with a velocity of 2 space units per time step. We consider a time step of 0.1 seconds and, at every time step, we update the position in space of each agent in the swarm. Since agents are asynchronous, we update their task allocation only when they execute a control rule. Agents are always assumed to work on one of the two tasks and to periodically execute their agent controller. Independently of the controller $\langle G; b\rangle$, agents perform a random walk; they do not collide with each other but can collide with the boundaries of the environment. In the case of a collision with a boundary, the agent bounces back with a mirrored angle of incidence. When executing the switch-or-recruit rule, agents note their own task allocation and sample the task allocations of their $G-1$ closest neighbors. In the following, we show the average of 10^4 simulations each lasting 10^5 seconds. Video recordings of the simulations can be found in the supplementary material[1].

4. GLOBAL-TO-LOCAL DESIGN METHOD

We build on the idea of behavioral heterogeneity to define a global-to-local design method for ST-MR task allocation problems. In doing so, we leverage on the (global-level) degrees of freedom resulting from mixing (at the local level) different agent controllers. Contrary to local-to-global approaches that manually explore a possibly infinite space of design solutions, we restrict our design-space and systematize our search for a solution. The outcome of this search—called *swarm composition*—is a heterogeneous swarm formed of groups of agents with different controllers.

Let $C = \{(\langle G_1; b_1\rangle, c_1), \ldots, (\langle G_m; b_m\rangle, c_m)\}$ represent a swarm composition with m agent controllers. For an agent controller $\langle G_i; b_i\rangle$, the value of c_i gives the number of agents in the swarm with that controller. We consider the problem of finding a swarm composition C that approximates the stationary distribution $\boldsymbol{\pi}$ defined by the user. We tackle

[1]The supplementary material is available at: `https://www.dropbox.com/s/z6g3xfzxdascsqo/SM.zip?dl=0`

this problem with a prescriptive model-driven approach by defining a macroscopic model that, given the local agent controllers, describes both the transient and the stationary behavior of the swarm; and a method to derive, from the desired stationary behavior $\boldsymbol{\pi}$, a (reference) model of a transient swarm behavior that converges to $\boldsymbol{\pi}$. The first model provides us with a space $\boldsymbol{E}$ of basis vectors, each describing an agent controller; the second model gives a response vector $\boldsymbol{y}$ that describes the behavior of the desired swarm. Finally, from a linear combination $\boldsymbol{y} = \boldsymbol{E}\boldsymbol{\beta}$ of basis vectors, we obtain the desired swarm composition C, $c_i \propto \boldsymbol{\beta}_i$.

4.1 Basis Vectors

For each agent controller, we use a discrete-time Markov chain $\{X(t) \in \mathcal{X} : \forall t \geqslant 0\}$ to describe the global dynamics of N agents executing the same controller. In the derivation of the Markov chain, we assume that at each time step one agent in the swarm executes a control rule. Note that this assumption does not lead to a slowdown of the allocation dynamics nor to any other loss in performance. Since agents act in real time—which is continuous—and are not synchronized, we have that the probability of two or more concurrent executions of control rules by different agents is zero[2]. The discretization of the time in terms of the number of control rule executions allows us to simplify our mathematical derivations without introducing approximations. In Section 5.2, we provide means to recover the time as a continuous entity from the number of control rule executions. A direct consequence of this assumption is that the number X of agents in the swarm allocated to task 0 changes by $\Delta \in \{+1, 0, -1\}$ units per time step. Therefore, the resulting Markov process is described by a tridiagonal matrix $\boldsymbol{P}_{G,b}$ of size $(N+1) \times (N+1)$.

For a given agent controller $\langle G; b\rangle$ with function $\Delta_{G,b} = (\Delta_i, \ldots, \Delta_{G-1})$, the transition matrix $\boldsymbol{P}_{G,b}$ is defined as

$$\boldsymbol{P}_{G,b}(X, X+1) = p_\sigma \left(1 - \frac{X}{N}\right) + p_\rho \sum_{k:\Delta_k=+1} \frac{\binom{X}{k}\binom{N-X}{G-k}}{\binom{N}{G}}, \quad (1)$$

$$\boldsymbol{P}_{G,b}(X, X) = p_\rho \sum_{k\in\{0,G\}} \frac{\binom{X}{k}\binom{N-X}{G-k}}{\binom{N}{G}}, \quad (2)$$

$$\boldsymbol{P}_{G,b}(X, X-1) = p_\sigma \frac{X}{N} + p_\rho \sum_{k:\Delta_k=-1} \frac{\binom{X}{k}\binom{N-X}{G-k}}{\binom{N}{G}}. \quad (3)$$

In Eq. (1), probability $\boldsymbol{P}_{G,b}(X, X+1)$ is the sum of two contributions: the probability that an agent allocated to task 0 self-switches its allocation to task 1; and the probability that any agent increases the value of X by applying the switch-or-recruit rule with a change $\Delta_k = +1$. The occurrence of a certain group composition is modeled using the hypergeometric distribution. In Eq. (2), $\boldsymbol{P}_{G,b}(X, X)$ results from those agents that do not execute any action as a result of the application of the switch-or-recruit rule over a homogeneous group with either 0 or G allocations for task 0. Eq. (3) is derived similarly to Eq. (1).

Eqs. (1–3) define an ergodic Markov chain $\boldsymbol{P}_{G,b}$. Based on $\boldsymbol{P}_{G,b}$, we derive a basis vector $\boldsymbol{e}_{G,b}$ that gives the expected change $\boldsymbol{e}_{G,b}(X)$ of the swarm allocation X resulting from the next agent executing a control rule. We obtain

$$\boldsymbol{e}_{G,b}(X) = +1 \cdot \boldsymbol{P}_{G,b}(X, X+1) - 1 \cdot \boldsymbol{P}_{G,b}(X, X-1). \quad (4)$$

Given a state X, the function $\boldsymbol{e}_{G,b}(X)$ returns the expected change $1/T \sum_{t=0}^{T\to\infty} X(t+1) - X(t)$ of X. We obtain an arbitrary large space $\boldsymbol{E}$ by considering all basis vectors $\boldsymbol{e}_{G,b}$, $b \in \{1, \ldots, 2^{G-1}\}$, for groups of increasing size G.

Figure 1a shows examples of basis vectors (dashed lines) and their linear combination (solid line). For $\boldsymbol{e}_{G,b}(X) > 0$ (respectively, $\boldsymbol{e}_{G,b}(X) < 0$), the transient behavior of the swarm drives the swarm allocation process towards the extreme allocations (either $X = 0$ or $X = N$). The zeros $e_{G,b}(X) = 0$ represent points that either attract or repel the swarm allocation process. Attraction points identify the modes of the stationary distribution $\boldsymbol{\pi}_{G,b}$ of $\boldsymbol{P}_{G,b}$. Basis vectors are pairwise symmetric with each other around $p_\sigma(1 - X/N)$, $X \in \mathcal{X}$; that is, for each b exists b' such that $e_{G,b} = p_\sigma(1 - X/N) - e_{G,b'}$. Therefore, an equal number of agents with controllers $\langle G; b\rangle$ and $\langle G; b'\rangle$ cancel each other's effect of the switch-or-recruit rule and leave only the contribution of the self-switching rule.

4.2 Response Vector

In our global-to-local design method, we obtain the response vector $\boldsymbol{y}$, which represents the expected change of the user-desired swarm, from the stationary distribution $\boldsymbol{\pi}$. To do so, we first construct a Markov chain $\boldsymbol{P}_y$ that converges to $\boldsymbol{\pi}$ itself and then compute $\boldsymbol{y}$ from $\boldsymbol{P}_y$ with Eq. (4).

The stationary distribution $\boldsymbol{\pi}$ of an ergodic Markov chain with transition matrix $\boldsymbol{P}$ can be uniquely determined by solving the system of equations $\boldsymbol{\pi}\boldsymbol{P} = \boldsymbol{\pi}$ [19]. The inverse problem, however, is less trivial and its solution is in general not unique. In the case of our tridiagonal transition matrix, this problem implies the exploration of a manifold $\{\boldsymbol{P}\}$ characterized by $2N$ dimensions. This number of dimensions is due to the sparse structure of tridiagonal matrices and to the fact that transition matrices are row-stochastic (i.e., row entries are non-negative and sum up to 1). As a consequence, in order to construct our response vector $\boldsymbol{y}$ we need to find a set of $2N$ additional constraints.

The stationary distribution $\boldsymbol{\pi}$ defined by the user imposes a set of $N+1$ linear constraints on this manifold through equation

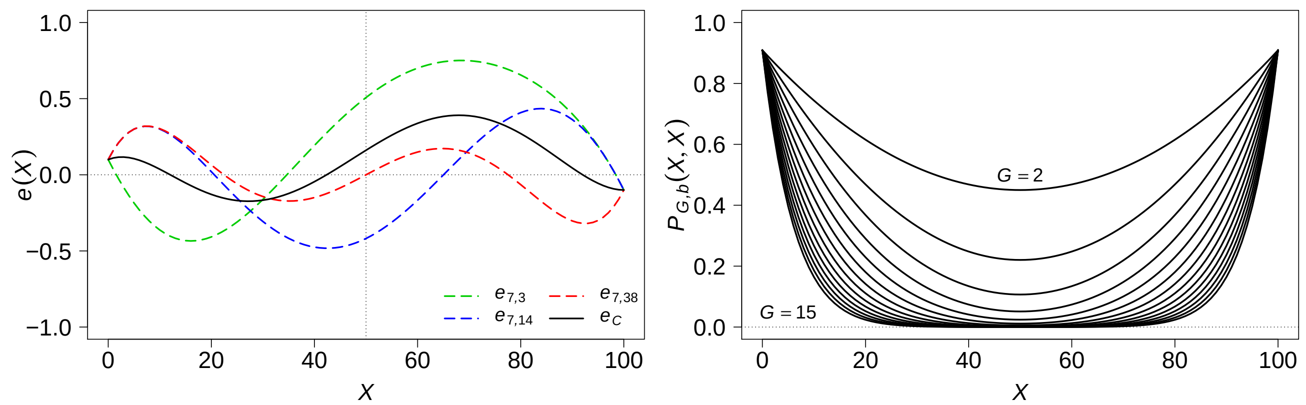
$$\boldsymbol{\pi}_i = \sum_{j\in\mathcal{X}} \boldsymbol{\pi}_j \boldsymbol{P}(j, i), \; \forall i \in \mathcal{X}.$$

Due to the linear relation $\sum_{i\in\mathcal{X}} \boldsymbol{\pi}_i = 1$ one of these constraints is redundant and the stationary distribution $\boldsymbol{\pi}$ reduces the number of dimensions of $\{\boldsymbol{P}\}$ from $2N$ to N. Therefore, a general transition matrix $\boldsymbol{P}_y$ that converges to $\boldsymbol{\pi}$ can be parameterized by N constant values referred to as $\boldsymbol{\psi} = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_N)$. By constraining the transition matrix $\boldsymbol{P}_y$ to be row-stochastic we obtain the set of inequalities

$$\begin{aligned} 0 \leqslant \; & \boldsymbol{\psi}_1\boldsymbol{\pi}_2 && \leqslant 1, \\ 0 \leqslant \; & \boldsymbol{\psi}_{i-2}\boldsymbol{\pi}_{i-2} + \boldsymbol{\psi}_{i-1}\boldsymbol{\pi}_i && \leqslant 1, \; \forall i \in \{3, N-1\}, \\ 0 \leqslant \; & \boldsymbol{\psi}_N\boldsymbol{\pi}_N && \leqslant 1. \end{aligned} \quad (5)$$

Any choice of values for parameters $\boldsymbol{\psi} = (\boldsymbol{\psi}_0, \ldots, \boldsymbol{\psi}_{n-2})$ that satisfies the above set of inequalities defines a transition matrix $\boldsymbol{P}_y$ that satisfies $\boldsymbol{\pi}\boldsymbol{P}_y = \boldsymbol{\pi}$. Since probabilities $\boldsymbol{\pi}_i$, $i \in \mathcal{X}$, are non-negative by definition, all entries in the parameter vector $\boldsymbol{\psi}$ can always be chosen to be sufficiently

[2] As a consequence of the time step of 0.1 seconds set in our multi-agent simulations and to the choice of values for parameters σ and ρ, we rarely observed concurrent executions of control rules by two or more agents.



(a) (b)

Figure 1: Illustration of a) basis vectors resulting from agent controllers $\langle 7;3\rangle, \langle 7;14\rangle, \langle 7;38\rangle$ and their linear combination $C = \{(\langle 7;3\rangle, 40), (\langle 7;14\rangle, 10), (\langle 7;38\rangle, 50)\}$ (parameters: $N = 100$, $\rho = 1$ and $\sigma = 1/9$); and b) the probability $\boldsymbol{P}_{G,b}(X,X)$ of not changing the current swarm allocation after the execution of the switch-or-recruit rule by an agent for increasing values of the group size G (parameters: $N = 100$, $\rho = 1$, $\sigma = 0.1$, $b = 1$, and $G \in \{2,\ldots,15\}$).

small to satisfy the set of inequalities in Eq. (5). Using Eq. (5) we have obtained N of $2N$ constraints necessary to determine a transition matrix $\boldsymbol{P}_y$ that asymptotically converges to $\boldsymbol{\pi}$.

In order to uniquely determine a transition matrix $\boldsymbol{P}_y$, we still require N additional constraints. By inspecting Eq. (2), we see that all agent controllers $\langle G;b\rangle$, $b \in \{1,\ldots,2^{G-1}\}$, have equal diagonal entries $\boldsymbol{P}_{G,b}(X,X)$. Furthermore, the probabilities $\boldsymbol{P}_{G,b}(X,X)$ converge for increasing values of the group size G as shown in Figure 1b. This implies that, by making an initial guess for parameters G, ρ, and σ, we can impose an additional set of $N+1$ linear constraints and uniquely determine a matrix $\boldsymbol{P}_y$. As we will see in the following, this initial guess of parameters is not binding and can be revised during the application of the method.

For a desired stationary distribution $\boldsymbol{\pi}$ and initial parameters G, ρ, and σ, we can solve $\boldsymbol{\pi}\boldsymbol{P}_y = \boldsymbol{\pi}$ and obtain the transition matrix $\boldsymbol{P}_y$. The solution of the system of equations is subject to two constraints: the diagonal entries of $\boldsymbol{P}_y$ are constant and equal to $\mathrm{diag}(\boldsymbol{P}_y) = \mathrm{diag}(\boldsymbol{P}_{G,b})$ (for any choice of $b \in \{1,\ldots,2^{G-1}\}$); and all rows of $\boldsymbol{P}_y$ are non-negative and sum up to 1. Since the first and last rows of $\boldsymbol{P}_y$ have only two non-zero entries, these two constraints suffice to compute $\boldsymbol{P}_y(0,1)$ and $\boldsymbol{P}_y(N,N-1)$. We compute all remaining entries $\boldsymbol{P}_y(X,X-1)$ and $\boldsymbol{P}_y(X,X+1)$ recursively following the sequence

$$\boldsymbol{P}_y(1,0) = \boldsymbol{\pi}_0 \frac{1-\boldsymbol{P}_y(0,0)}{\boldsymbol{\pi}_1}, \tag{6}$$

$$\boldsymbol{P}_y(1,2) = 1 - \boldsymbol{P}_y(1,1) - \boldsymbol{P}_b(1,0), \tag{7}$$

$$\cdots$$

$$\boldsymbol{P}_y(X,X-1) = \boldsymbol{\pi}_{X-1} \frac{1-\boldsymbol{P}_y(X-1,X)}{\boldsymbol{\pi}_X}, \tag{8}$$

$$\boldsymbol{P}_y(X,X+1) = 1 - \boldsymbol{P}_y(X,X) - \boldsymbol{P}_y(X,X-1). \tag{9}$$

Finally, the response vector $\boldsymbol{y}$ is obtained from $\boldsymbol{P}_y$ by computing its expected change as in Eq. (4).

4.3 Regression Problem

Starting from an arbitrary set $\mathcal{B} = \{\langle G_1;b_1\rangle, \langle G_2;b_2\rangle,\ldots\}$ of agent controllers, Eq. (4) allows us to define our search space using a matrix $\boldsymbol{E}$ whose columns are the transposed basis vectors $\boldsymbol{e}_{G,b}$, $\langle G;b\rangle \in \mathcal{B}$. The response vector $\boldsymbol{y}$ is derived from the stationary distribution $\boldsymbol{\pi}$ using Eqs. (6–9) and Eq. (4). In order to determine our swarm composition C, we need to find a column vector $\boldsymbol{\beta}$ of regression coefficients that satisfies

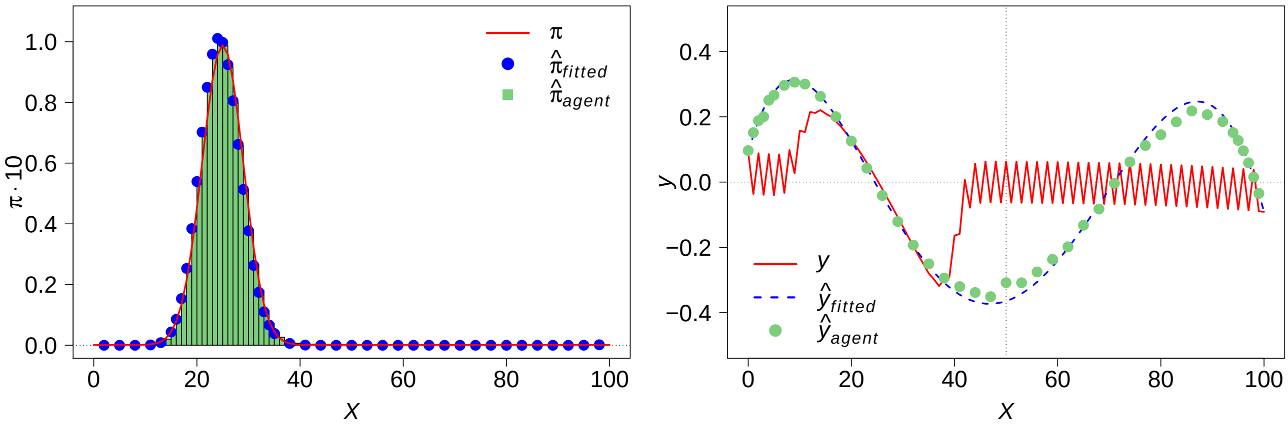
$$\boldsymbol{y} \simeq \boldsymbol{E}\boldsymbol{\beta},\ \ \boldsymbol{\beta}_i \geqslant 0. \tag{10}$$

Coefficients $\boldsymbol{\beta}_i$ are required to form a conical combination, i.e., $\boldsymbol{\beta}_i \geqslant 0$, so that $c_i \simeq N\boldsymbol{\beta}_i$ results in a non-negative number of agents with controller $\langle G_i;b_i\rangle$.

In general, the accuracy of a solution to the regression problem in Eq. 10 increases with the number of basis vectors whose coefficient $\boldsymbol{\beta}_i$ is greater than zero. However, this scenario implies the use of many different agent controllers which might compromise the robustness of the designed swarm. In fact, a swarm composition based on many different agent controllers is more affected by agent failures because each agent controller is likely to be represented by only a few agents in the swarm. As a result, in case of agent failures, the actual swarm composition might soon depart from the designed one. In contrast, a swarm with few agent controllers but many agents for each of them suffers less from the loss of agents because these losses are more likely to be homogeneously distributed across agent controllers. In this case, the swarm will still allocate its agents as specified by the user input. In order to increase the robustness of the designed swarm composition, we seek to maximize the number of agents using each of the selected agent controller and therefore to minimize the number of agent controllers used in the designed swarm composition.

We therefore search for a solution that minimizes the number of non-zero coefficients $\boldsymbol{\beta}_i$. We obtain this objective by defining the regression problem as a lasso problem [36] with positivity constraints

$$\underset{\hat{\boldsymbol{\beta}}\in\mathbb{R}^n}{\arg\min}\ \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{E}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1,\ \boldsymbol{\beta}_i \geqslant 0,\ \forall i \in \mathcal{X}. \tag{11}$$



(a) (b)

Figure 2: Illustration of the design method and comparison with multi-agent simulations for the unimodal scenario: a) depicts the stationary distribution and b) the expected change.

In the above equation, the regularization coefficient λ determines the weight of the ℓ_1-penalization term and controls the sparsity of the solution $\hat{\boldsymbol{\beta}}$. Given a solution $\hat{\boldsymbol{\beta}}$ of (11), we normalize each coefficient $\boldsymbol{\beta}_i$ according to $\boldsymbol{\beta}_i = \boldsymbol{\beta}_i / \sum_{j=1}^{m} \boldsymbol{\beta}_j$ so that the coefficients $\boldsymbol{\beta}_i$ sum to 1 and satisfy the physical conservation of the swarm size. The final swarm composition C is obtained by computing the number c_i of agents with controller $\langle G_i; b_i \rangle$ as $c_i = N\boldsymbol{\beta}_i$ and rounding these values in order to have integer numbers of agents for each controller and a swarm of size N.

5. DESIGN OF TASK ALLOCATION

We apply our method to design heterogeneous swarms for both unimodal and multimodal user inputs $\boldsymbol{\pi}$. As introduced above, heterogeneous swarms formed by many different agent controllers might not be robust to failures of individual agents. Therefore, we aim at minimizing the number of agent controllers in the designed swarms. In doing so, we give priority to the robustness of the designed solution and prefer qualitative over quantitative accuracy in the approximation of $\boldsymbol{\pi}$. In the following, we design swarms with $N = 100$ agents. Since $\boldsymbol{\pi}$ is independent of the magnitude of ρ and σ, but only depends on probabilities p_ρ and p_σ, we set $\rho = 1$ and vary σ in $[0; 1]$. In the multi-agent simulations, ρ and σ are divided by a factor of 10^2.

5.1 Unimodal User Input

Figures 2a and 2b show the results of the proposed method applied to a unimodal user input. The red solid line in Figure 2a represents the user input $\boldsymbol{\pi}$ which defines the desired allocation $(25, 75)$. From $\boldsymbol{\pi}$, we derive a response vector $\boldsymbol{y}$ by first constructing an equivalent Markov chain as in Eqs. (6–9) and successively applying Eq. 4. To this end, we initially set parameters to $G = 6$ and $\sigma = 0.1$. The resulting response vector $\boldsymbol{y}$ (red solid line in Figure 2b) shows sudden jumps for values of $X \notin [10; 40]$. These jumps can be reduced by tuning the initial values of G and σ. However, we observe that tuning is not necessary and might even worsen the accuracy of our design method.

We consider asymmetric agent controllers for $G \in \{3, \dots, 6\}$ and solve the lasso problem in (11) for $\lambda = 1$. We obtain the swarm composition $C_1 = \{(\langle 6; 7\rangle, 39), (\langle 6; 11\rangle, 5), (\langle 6; 15\rangle, 56)\}$ that consists of 3 agent controllers with $G = 6$. Due to the requirement of sparsity, the expected change $\hat{\boldsymbol{y}}_{\text{fitted}}$ computed from C_1 using the Markov chain does not accurately fit the response vector $\boldsymbol{y}$ (see Figure 2b). This also applies to the expected change $\hat{\boldsymbol{y}}_{\text{agent}}$ that results from multi-agent simulations. Nonetheless, the designed solution reproduces its essential features (i.e., the zeros and the signs of $\boldsymbol{y}$) in the region of interest (i.e, in the range $[10; 40]$). This suffices to design a composition C_1 that closely meets the user input as shown in Figure 2a by the distribution $\hat{\boldsymbol{\pi}}_{\text{fitted}}$ predicted using both the Markov chain model (blue circles) and the distribution $\hat{\boldsymbol{\pi}}_{\text{agent}}$ resulting from multi-agent simulations (histograms).

Similarly to the solution proposed in [3], our method can also be used to implement sequential task allocation. Let us consider a series of user inputs $\boldsymbol{\pi}^1, \dots, \boldsymbol{\pi}^k$. By applying our method to each user input we can derive a set of swarm compositions $\{C_1, \dots, C_k\}$. Individual agents in the swarm could be programmed to change their controller over time according to $\{C_1, \dots, C_k\}$. Depending on the scenario, the change of agent controllers can be coupled to external signals broadcast by the designer, a predefined time schedule, or changing environmental cues. We performed a simple experiment where the agents in the swarm change their agent controller after a certain predefined time. Initially, the swarm is required to allocate its agents around the swarm allocation $(25, 75)$ as specified by the distribution $\boldsymbol{\pi}^1 = \boldsymbol{\pi}$ given in Figure 2a and uses the swarm composition $C_1^1 = \{(\langle 6; 7\rangle, 39), (\langle 6; 11\rangle, 5), (\langle 6; 15\rangle, 56)\}$. In a second time period, the swarm is required to change the distribution. The second distribution $\boldsymbol{\pi}^2$ over swarm allocations (not shown here) defines the swarm allocation $(75, 25)$ and is obtained by the swarm composition $C_1^2 = \{(\langle 6; 19\rangle, 59), (\langle 6; 20\rangle, 41)\}$. We show the feasibility of this approach with a video recording of this simulation provided in the supplementary material (see Footnote 1). Agents are initialized using the swarm composition C_1^1 and readily converge to $\boldsymbol{\pi}^1$. At time $t = 500$ seconds, the agents in the swarm change their agent controllers from the initial swarm composition C_1^1 to the second swarm composition C_1^2; soon after the change of agent controllers, the swarm updates its allocation and converges to $\boldsymbol{\pi}^2$.

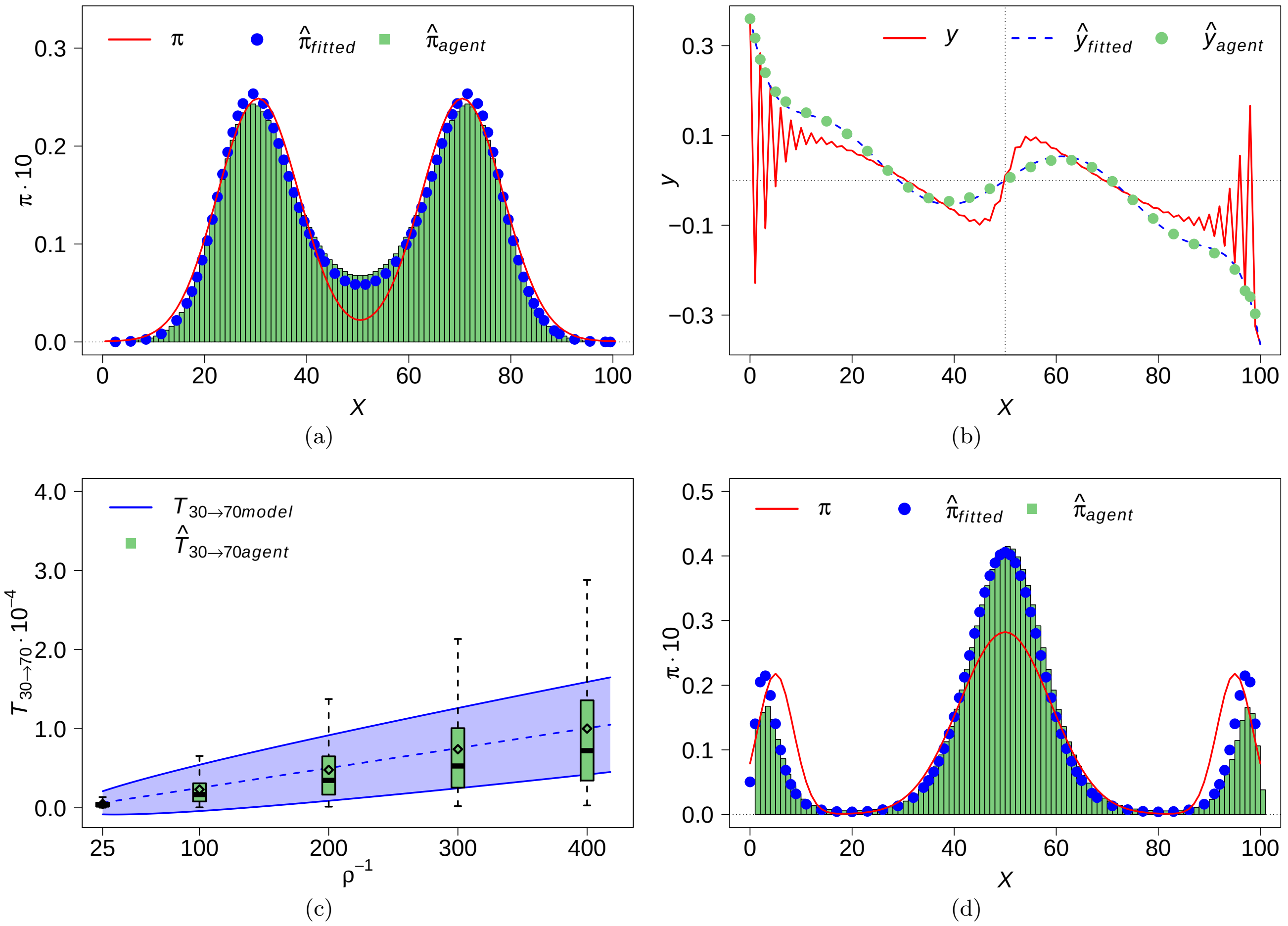


Figure 3: Illustration of the design method and comparison with multi-agent simulations. For the bimodal scenario a) depicts the stationary distribution, b) the expected change, and c) the mean switching time. Figure d) depicts the stationary distribution of the trimodal scenario.

5.2 Multimodal User Input

A multimodal stationary distribution $\boldsymbol{\pi}$ defines a task allocation scenario characterized by multiple swarm allocations $(X_1, N - X_1), \ldots, (X_k, N - X_k)$ (one for each mode of $\boldsymbol{\pi}$). The result of such a user input is a swarm that periodically switches between pairs of swarm allocations. Contrary to the above discussed case, switches between pairs of swarm allocations are stochastic and characterized by a certain mean period of time (see [8] for a bimodal example). Thus, multimodal user inputs define a periodic task allocation scenario.

Figures 3a and 3b show an example application of our method to a bimodal user input. The red solid line in Figure 3a defines a scenario with two swarm allocations: $(30, 70)$ and $(70, 30)$. The response vector $\boldsymbol{y}$ (red line in Figure 3b) has been derived using initial parameters $G = 6$ and $\sigma = 0.575$. We consider all agents controllers $\langle G; b\rangle$ resulting from group sizes $G \in \{3, \ldots, 9\}$ and solve the lasso problem for $\lambda = 3$. The solution of Eq. (11) yields the swarm composition $C_2 = \{(\langle 9; 78\rangle, 92), (\langle 9; 141\rangle, 4), (\langle 9; 207\rangle, 4)\}$ characterized by 3 agent controllers with group size $G = 9$. With respect to the unimodal scenario, we increased the value of the regularization parameter λ to obtain a sparse solution $\hat{\boldsymbol{\beta}}$. Both the stationary distribution $\hat{\boldsymbol{\pi}}_{\text{fitted}}$ predicted using the Markov chain and the distribution $\hat{\boldsymbol{\pi}}_{\text{agent}}$ computed from multi-agent simulations (shown in Figure 3a) qualitatively match the user input $\boldsymbol{\pi}$ with only a small deviation of the distribution around $X \in [45; 55]$. A video recording of a multi-agent simulation of the bimodal scenario can be found in the supplementary material (see Footnote 1).

Additionally, the user might also express requirements over the mean switching time $T_{X_1 \to X_2}$, that is, the time necessary for the swarm to reallocate its agents from $(X_1, N - X_1)$ to $(X_2, N - X_2)$. Using the Markov chain model resulting from C_2, we can compute the mean and the variance of the number of control rule executions necessary for this purpose. By multiplying these statistics by the mean duration $\rho^{-1}N + \sigma^{-1}N$ between the execution of two control rules we can obtain $T_{30\to70}$ as a function of the rates ρ and σ. In doing so, we recover the time in its continuous form from the discrete number of executions of control rules. Figure 3c shows the prediction of the Markov chain ($T_{30\to70\text{model}}$, shaded area) compared to multi-agent simulations ($\hat{T}_{30\to70\text{agent}}$ box-plots) when $\sigma = 0.575\rho$ and $\rho^{-1} \in [25; 420]$. We obtain a good agreement of both the means (dashed line versus diamonds symbols) and the variance of the two models.

Finally, we apply our global-to-local design method to a trimodal user input $\boldsymbol{\pi}$. The stationary distribution $\boldsymbol{\pi}$ (red line in Figure 3d) defines a task allocation scenario where the swarm alternates its workforce among 3 possible allocations: $(10, 90)$, with the majority of agents working on task 1; $(50, 50)$ with agents equally allocated to both tasks; and $(90, 10)$ with the majority of agents working on task 0. We compute the response vector $\boldsymbol{y}$ using initial parameters $G = 5$ and $\sigma = 0.2$ (data not shown). We define the minimization problem in Eq. (11) by considering only asymmetric agent controllers with $G = 7$ (i.e., the first 32 basis vectors). The solution of the lasso problem for $\lambda = 1$ gives the swarm composition $C_3 = \{(\langle 7; 22\rangle, 56), (\langle 7; 26\rangle, 44)\}$. The distribution $\hat{\boldsymbol{\pi}}_{\text{fitted}}$ predicted using the Markov chain and the distribution $\hat{\boldsymbol{\pi}}_{\text{agent}}$ resulting from simulations qualitatively recover the user input (blue circles and histograms).

6. DISCUSSION

In the previous section, we have shown that our method can be used to design swarms that allocate their agents as defined by the user input. The user input is a stationary probability distribution over swarm allocations and defines the probability of any possible swarm allocation. Our method allows the user to specify scenarios with a single swarm allocation using a unimodal distribution and scenarios where the swarm alternates between different swarm allocations using a multimodal distribution.

A unimodal stationary distribution $\boldsymbol{\pi}$ defines a task allocation scenario with a single swarm allocation $(X, N - X)$. In principle, this scenario could be tackled by a static assignment of agents to each of the two tasks. However, such an approach is not robust to individual agent failures. Consider for example a collective construction scenario where task 0 and task 1 require, respectively, to dig and to remove the excavation material from a construction site. Due to workload disparity between tasks, agents are likely to experience uneven failure rates. Over time, the swarm might significantly depart from the desired allocation $(X, N - X)$. Without complete knowledge of individual agent failures, a designer would be prevented from restoring the initial static allocation (e.g., by deploying new agents). In contrast, our design method is robust to this situation. Since agents repeatedly switch tasks, the workload is shared equally among agents. Agents are thus equally subject to failures and the desired proportions of agents with each controller is preserved. In this scenario, the swarm will still allocate as defined by the user. Moreover, upon degradation of the swarm performance, the designer can add new agents to the system with the same proportion of agent controllers as originally designed. Note that the addition of new agents in the swarm can be used also as a mean to reprogram the swarm behavior by considering the original swarm composition as an additional constraint in our design method.

As discussed in Section 5.1, we can use a sequence of unimodal distributions to design sequential task allocation scenario. This is achieved by letting agents change their agent controllers over time and results in a swarm that moves from a swarm allocation to the successive in the sequence. Our method can be used to design a swarm composition for each distribution in the sequence; however, this approach to sequential task allocation requires agents with a mechanism (e.g., based on an external signal or fixed time scheduling) that triggers the change of agent controller.

Alternatively, the user might provide a multimodal distribution as input. In this case, with a single swarm composition that does not change over time, we obtain a swarm behavior that naturally oscillates between the swarm allocations defined by the modes of the user input. This type of self-organizing swarm behavior is similar to the one recently investigated by Silk et al. for the design of self-organizing networks [35]. Periodic task allocation offers an alternative approach to implement sequential task allocation which might be useful in extreme applications where hardware limitations might prevent agents from perceiving external signals or from being freely programmable (e.g., hard-wired controllers in nanorobotics applications [7]). This alternative approach to sequential task allocation could be useful, for example, to increase the penetration of nanobots into tumors [**?**]. Nanobots with multifunctional capabilities (e.g., sensing, imaging, therapy) [34] could be designed to initially perform tissue penetration and diagnosis and later to deliver the drugs in their payloads.

7. CONCLUSION

We introduced the idea that swarms that achieve user-specified objectives can be designed by leveraging on behavioral heterogeneity. This idea was inspired by the concept of population coding from neurosciences [30], where a population of neurons performs a function approximation by combining different heterogeneous contributions. We explored this new design paradigm by defining a global-to-local design method for self-organized binary task allocation. Our method works by solving a penalized regression problem to select which agent controllers should be used to approximate the user input. We have shown that this method can be successfully used to design swarms for static, sequential (i.e., unimodal distributions), and periodic binary task allocation (i.e., multimodal distributions). In current research we are trying to understand why the selection process seems to be biased towards agent controllers with large group size G—in certain scenarios small values of this parameter might be more desirable.

In future research we intend to extend the method so that it can be applied to task allocation scenarios with more than two tasks. This extension will require the definition of other linear constraints in addition to those defined in Section 4.2 that are necessary to uniquely derive a response vector from the user input. This could be achieved, for example, by considering different priorities among the tasks to be executed. We note that the number of agent controllers is an exponential function of the number of tasks. However, penalized regression techniques allow us to consider high-dimensional search spaces and to investigate a reasonable range of application scenarios. We also plan to perform a thorough algebraic characterization of our basis vectors and response vector with the aim to improve the performance of the design method. We believe that our idea of combining behavioral heterogeneous agents by means of prescriptive model-based approaches has potential for applications beyond the example of task allocation considered in this paper. Our primary goal is therefore to deepen our understanding of the fundamental principles of behavioral heterogeneity (i.e., to answer questions such as: which features allow to combine agent controllers?) with the aim to extend our approach to swarm scenarios different from task allocation (e.g., collective decision-making or spatially organizing tasks).

REFERENCES

[1] W. Agassounon and A. Martinoli. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '02, pages 1090–1097. ACM, 2002.

[2] W. Agassounon, A. Martinoli, and K. Easton. Macroscopic modeling of aggregation experiments using embodied agents in teams of constant and time-varying sizes. *Autonomous Robots*, 17(2-3):163–192, 2004.

[3] S. Berman, A. Halasz, M. Hsieh, and V. Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, 25(4):927–937, 2009.

[4] S. Berman, R. Nagpal, and A. Halasz. Optimization of stochastic strategies for spatially inhomogeneous robot swarms: A case study in commercial pollination. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3923–3930, 2011.

[5] M. Brambilla, A. Brutschy, M. Dorigo, and M. Birattari. Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking. *ACM Trans. Auton. Adapt. Syst.*, 9(4):17:1–17:28, 2014.

[6] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[7] D. Bray. Protein molecules as computational elements in living cells. *Nature*, 376(6538):307–312, 07 1995.

[8] J. Buhl, D. J. T. Sumpter, I. D. Couzin, J. J. Hale, E. Despland, E. R. Miller, and S. J. Simpson. From disorder to order in marching locusts. *Science*, 312(5778):1402–1406, 2006.

[9] M. Castillo-Cagigal, A. Brutschy, A. Gutiérrez, and M. Birattari. Temporal task allocation in periodic environments. In *Swarm Intelligence*, volume 8667 of *LNCS*, pages 182–193. Springer, 2014.

[10] J. Cheng, W. Cheng, and R. Nagpal. Robust and self-repairing formation control for swarms of mobile agents. In *Proceedings of the Twentieth AAAI Conference on Artificial Intelligence*, volume 5, pages 59–64. AAAI Press, 2005.

[11] K. Dantu, S. Berman, B. Kate, and R. Nagpal. A comparison of deterministic and stochastic approaches for allocating spatially dependent tasks in micro-aerial vehicle collectives. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 793–800, 2012.

[12] M. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[13] M. Dorigo, M. Birattari, and M. Brambilla. Swarm robotics. *Scholarpedia*, 9(1):1463, 2014.

[14] A. Dussutour, M. Beekman, S. Nicolis, and B. Meyer. Noise improves collective decision-making by ants in dynamic environments. *Proceedings of the Royal Society B: Biological Sciences*, 276(1677):4353–61, 2009.

[15] A. Georgopoulos, A. Schwartz, and R. Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.

[16] B. P. Gerkey and M. J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[17] H. Hamann, G. Valentini, Y. Khaluf, and M. Dorigo. Derivation of a micro-macro link for collective decision-making systems: Uncover network features based on drift measurements. In *Parallel Problem Solving from Nature – PPSN XIII*, volume 8672 of *LNCS*, pages 181–190. Springer, 2014.

[18] H. Hamann and H. Wörn. A framework of space-time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, 2(2-4):209–239, 2008.

[19] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer, 1976.

[20] D. Kengyel, H. Hamann, P. Zahadat, G. Radspieler, F. Wotawa, and T. Schmickl. Potential of heterogeneity in collective behaviors: A case study on heterogeneous swarms. In Q. Chen, P. Torroni, S. Villata, J. Hsu, and A. Omicini, editors, *PRIMA 2015: Principles and Practice of Multi-Agent Systems*, volume 9387 of *LNCS*, pages 201–217. Springer, 2015.

[21] E. Klavins. Programmable self-assembly. *Control Systems, IEEE*, 27(4):43–56, Aug 2007.

[22] P. Kouvaros and A. Lomuscio. A counter abstraction technique for the verification of robot swarms. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2081–2088. AAAI Press, 2015.

[23] P. Kouvaros and A. Lomuscio. Verifying emergent properties of swarms. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1083–1089. AAAI Press / IJCAI, 2015.

[24] M. J. B. Krieger, J.-B. Billeter, and L. Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406:992–995, Aug 2000.

[25] T. H. Labella, M. Dorigo, and J.-L. Deneubourg. Division of labor in a group of robots inspired by ants' foraging behavior. *ACM Trans. Auton. Adapt. Syst.*, 1(1):4–25, 2006.

[26] K. Lerman, A. Martinoli, and A. Galstyan. A review of probabilistic macroscopic models for swarm robotic systems. In *Swarm Robotics*, volume 3342 of *LNCS*, pages 143–152. Springer, 2005.

[27] K. H. Low, W. K. Leow, and M. H. Ang Jr. Task allocation via self-organizing swarm coalitions in distributed mobile sensor network. In *Proceedings of the Nineteenth AAAI Conference on Artificial Intelligence*, volume 4, pages 28–33. AAAI Press, 2004.

[28] M. Massink, M. Brambilla, D. Latella, M. Dorigo, and M. Birattari. On the use of bio-pepa for modelling and analysing collective behaviours in swarm robotics. *Swarm Intelligence*, 7(2-3):201–228, 2013.

[29] R. Nagpal. Programmable self-assembly using biologically-inspired multiagent control. In *Proceedings of the First International Conference on Autonomous*

Agents and Multiagent Systems, AAMAS '02, pages 418–425. ACM, 2002.

[30] A. Pouget, P. Dayan, and R. Zemel. Information processing with population codes. *Nature Reviews Neuroscience*, 1(2):125–132, 11 2000.

[31] M. Rubenstein, A. Cornejo, and R. Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[32] E. Şahin. Swarm robotics: From sources of inspiration to domains of application. In E. Şahin and W. Spears, editors, *Swarm Robotics*, volume 3342 of *LNCS*, pages 10–20. Springer, 2005.

[33] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998.

[34] J. sil Choi, Y. wook Jun, S.-I. Yeon, H. C. Kim, J.-S. Shin, and J. Cheon. Biocompatible heterostructured nanoparticles for multimodal biological detection. *Journal of the American Chemical Society*, 128(50):15982–15983, 2006.

[35] H. Silk, M. Homer, and T. Gross. Design of self-organising networks. *ArXiv e-prints*, 1510.05045, Oct 2015.

[36] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Statist. Soc. B*, 58(1):267–288, 1996.

[37] G. Valentini, M. Birattari, and M. Dorigo. Majority rule with differential latency: An absorbing Markov chain to model consensus. In *Proceedings of the European Conference on Complex Systems 2012*, Springer Proceedings in Complexity, pages 651–658. Springer, 2013.

[38] G. Valentini, H. Hamann, and M. Dorigo. Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1305–1314. IFAAMAS, 2015.

[39] J. Werfel, K. Petersen, and R. Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014.

[40] C. A. Yates, R. Erban, C. Escudero, I. D. Couzin, J. Buhl, I. G. Kevrekidis, P. K. Maini, and D. J. T. Sumpter. Inherent noise can facilitate coherence in collective swarm motion. *Proceedings of the National Academy of Sciences*, 106(14):5464–5469, 2009.