Probabilistic Modeling of Programmable Stochastic Self-Assemby of Robotic Modules

Bahar Haghighat, Robin Thandiackal, Maximilian Mordig, and Alcherio Martinoli

Abstract— **Creating accurate models of stochastically self-assembling systems is a key step in developing control strategies, centralized or distributed, for the self-assembly process. This paper comparatively studies several aspects of developing probabilistic models for programmable self-assembling systems of stochastically interacting modules. In particular, we systematically investigate Markov models as well as hidden Markov models to predict the self-assembly process dynamics. We consider a case study leveraging our fluidic self-assembly robotic system. The ground truth is obtained through a high-fidelity simulation, calibrated using real experimental data. We first consider Markov models and employ the formalism of chemical reaction networks. In order to compute the model parameters, i.e. the reaction rates in the network, three different methods are studied. We then investigate the validity of the underlying well-mixed assumption, and thus the Markov property, for our system through estimation of the diffusion coefficient, through two different approaches. The system is shown to be borderline well-mixed, motivating extension of the initial Markov models to more complex models in order to achieve improved model accuracy. We formulate an automatic method for creating a hidden Markov model starting from a Markov model, based on a previously existing systematic method. Sample trajectories of the models are realized using the Gillespie's method. The resulting hidden Markov model is shown to achieve an improved accuracy over the standard Markov model.**

I. Introduction

Self-assembling robotic systems have garnered significant interest for their robust performances in forming structures of varied complexities and scales as well as their minimal design of constituting modules [1], [2], [3]. These systems employ Self-Assembly (SA) as a coordination mechanism among the modules to put together desired target structures of specific forms or functionalities in presence of typically highly noisy sensing, actuation, and interaction dynamics [4].

This paper studies different aspects of developing Markovian probabilistic models for a class of self-assembling robotic systems where the constituting modules are randomly stirred in a confined arena and interact to assemble structures based on their embedded ruleset controllers. For a given desired target structure, the engineering goal in programmable stochastic self-assembling systems is to derive and program proper ruleset controllers on the modules such that the target structure emerges in a reliable and predictable fashion [5]. The environmental features may also be controlled to assist the assembly of the target structure [6], [7].

A key component in studying programmable stochastic self-assembling systems is developing models that accurately describe the assembly process dynamics. Such models would help in: (1) accurately predicting the performances (assembly rate and yield) of the distributed system, and (2) evaluating and optimizing control strategies, whether distributed (e.g., ruleset controllers programmed on the modules) or centralized (e.g., modulating environmental features such as mixing forces deriving random interactions among modules), based on model predictions [8], [9].

Several works have addressed developing Markovian probabilistic models for stochastic self-assembling systems to date [1], [5], [6], [10], [11]. The choice of employing probabilistic modeling techniques for such systems is essentially motivated by the randomness lying at the core of these systems: random motion of the modules in the environment, explicit random decisions made by the modules' embedded controller, and random interactions among the modules [12]. Additionally, probabilistic models can be employed to provide a high-level macrostate description of the system state at each point in time by abstracting away low-level physical details of the system state such as positions, velocities, and internal states of all modules (i.e. microstate description). A general methodology for developing accurate probabilistic models of the dynamics of programmable self-assembling systems is sought after to date.

In this work, our focus is on creating a general approach for developing a discrete-state hidden Markov process model of programmable stochastic SA directly obtained from (1) a description of the robotic modules' embedded ruleset controller, and (2) an estimation of the rate constants defining the formation rates of different assemblies. To demonstrate this approach, we consider the case of our floating self-assembling robotic system, where the SA process is controlled in a fully distributed fashion through the programmable embedded ruleset controllers of the robotic modules. We use a high-fidelity calibrated simulation of the system as the ground truth and address different aspects of developing probabilistic macrostate models for the system. The contributions of this work are along three axes (1) a new rate estimation method for computing Markov model parameters is introduced and compared with two existing ones [1], [5], (2) a new method for estimating diffusion coefficient and evaluating the well-mixed condition is studied and compared with an existing one [5], and (3) a systematic

Bahar Haghighat, Maximilan Mording, and Alcherio Martinoli are with the Distributed Intelligent Systems and Algorithms Laboratory, School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne, Switzerland. firstname.lastname@epfl.ch

Robin Thandiackal is with the Biorobotics Laboratory, School of Engineering, École Polytechnique Fédérale de Lausanne, Switzerland. firstname.lastname@epfl.ch

This work has been sponsored by the Swiss National Science Foundation under the grant numbers 200021_137838/1 and 200020_157191/1.

method in the literature for refining Markov models using Hidden Markov Model (HMM) formalism, presented in [6], is employed to develop an HMM through automatic refinement of an initial Markov model of the system.

II. Related Work

Formally defined as the reversible and spontaneous phenomenon of an ordered spatial structure emerging from aggregate behavior of simpler preexisting entities, SA has been realized in engineered systems of both passive [6], [13], and active modules [14], [15]. Stochastic SA, demonstrated in [6], [7], [13], [14], is realized through taking advantage of the stochastic ambient energy for module transportation. This in turn allows for further simplification of the modules internal design. In [14], [15], programmable SA has been demonstrated where the shape of the forming target assembly is determined by the controller programmed on the robotic modules. Several applications are envisioned for programmable SA, for instance, programmable matter [16], space-deployed self-assembling robots [17], and novel manufacturing techniques at micro/nano-scales [18].

Diffusion derives the mixing in stochastically self-assembling systems. Under the assumption that the system is well-mixed, that is, the interacting modules are uniformly distributed in the system and that they diffuse faster than they react, the evolution of the system macrostate can be described by a Markov process whose dynamics is described by the Chemical Master Equation (CME). CME describes a population model of the system, where the state of the system is described by the copy number of each assembly configuration, i.e. chemical compound type, rather than the individual modules' positions and velocities. A compact description of the CME can be captured by the Chemical Reaction Network (CRN) formalism, where the CME is described from the modules perspective, i.e. by listing the states they can be in [6]. The parameters of the CRN model, i.e. the chemical reaction rate constants, can be computed by initializing the system in specific states and observing the time delay for specific reactions to occur [1], [10]. Having the Markov process model of the system, one can generate sample trajectories of the macrostate evolution using efficient exact and approximate algorithms [19]. However, the CME formulation's underlying assumption of well-mixed conditions is not always valid in real-world systems. The Markov model will thus fail to accurately predict the macrostates evolution. In order to improve model accuracy, several methods based on HMMs have been proposed to augment the system's original Markov model and its corresponding CME with hidden states [6], [10]. In [10], the authors study a system of programmable modules and propose to manually augment the original CRN species with the types of previous interaction partners, considering that two modules would be more likely to interact next if they have interacted recently. A systematic approach is studied in [6] for modeling a system of passive modules, where each species is augmented with hidden species based on a maximum likelihood approach regardless of the actual interpretation of the hidden species.

III. Markovian Models for Programmable Self-Assembly

We employ similar formulations as in [6] and [10], and provide formal definitions of several concepts in this section.

A. Markov Models

A discrete-time stochastic process can be defined as a collection of random variables, $\{X_n\}_{n\in\mathbb{N}}$. Similar to realization of a single random variable, a realized trajectory $\omega \in \mathbb{N} \to X$ generated by the stochastic process defines an assignment of the random variables X_n to particular values x in state space X. A discrete-time discrete-state Markov process, i.e. a Markov chain, has the following property, where $x_i \in X$:

$$Pr\{X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, ..., X_0 = x_0\} \\ = Pr\{X_{n+1} = x_{n+1} | X_n = x_n\} \quad (1)$$

The transition dynamics of a Markov chain is specified by the one-step transition probabilities, where matrix A is independent of n for a stationary process:

$$A_{ij}(n) = Pr\{X_{n+1} = j | X_n = i\} \quad for\ i, j \in X \qquad (2)$$

Consider a self-assembling robotic system consisting of N_0 individual robotic modules that may be in N_s different control states $q_1, \ldots, q_{N_s}$ as specified in their embedded ruleset controller. At the microscopic level, the state of the system can be described by the vector:

$$\vec{X}^{\text{micro}}(t) = \big[Q_1(t), Q_2(t), \ldots, Q_{N_0}(t)\big], \qquad (3)$$

where $Q_i(t) = q_1, \ldots, q_{N_s}$ is the state of module i at time t. The embedded ruleset controller guides the modules to build the target structure through building specific intermediate assembly configurations. Assuming that the set of possible assembly configurations as prescribed by the embedded ruleset controller is $c_1, \ldots, c_{N_c}$. At the macroscopic level, one is interested in the number of copies of the specific assembly configurations. Therefore, the state of the system can be described as:

$$\vec{X}^{\text{macro}}(t) = \big[N_1(t), N_2(t), \ldots, N_{N_c}(t)\big], \qquad (4)$$

where $N_i \in \mathbb{N}_{\geq 0}$ is the number of copies of assembly configuration c_i at time t. We consider discrete-time, with $\vec{X}_n^{\text{macro}}$ indicating the system macrostate at the n^{th} timestep. Assuming well-mixed conditions, the evolution of the system state may be expressed as a discrete-time discrete-state Markov process $\{\vec{X}^{\text{macro}}\}_{n\in\mathbb{N}}$. Note that in order to fully specify the model, one has to list all the feasible system macrostates as well as the transition probabilities between them.

B. Hidden Markov Models

An HMM can be defined as a stochastic process with state space Y whose trajectories are described by a Markov chain with state space X and an output function $f : X \to Y$. The function f can be many-to-one, therefore, the resulting process Y can be non-Markov while the process X is. In practice, HMMs correspond to the case where the system state is not fully observable and as a result several underlying Markov process states (hidden states) may correspond to the same observable state of a non-Markov process.

C. Chemical Reaction Networks

A CRN $\mathcal{N} = (\mathcal{R}, \mathcal{S})$ is a set of reactions $\mathcal{R} = \{R_1, \dots, R_{N_R}\}$ acting on a set of species $\mathcal{S} = \{S_1, \dots, S_{N_S}\}$. Each reaction R is then defined as two vectors of nonnegative integers specifying the stoichiometry of the reactants, $\vec{r}_R = [r_{R,1}, \dots, r_{R,N_S}]$, and the products species, $\vec{p}_R = [p_{R,1}, \dots, p_{R,N_S}]$, respectively. The stoichiometry determines the number of copies of a given reactant or product species that is required or produced when a reaction occurs. The CRN provides a population model, it thus keeps track of the number of copies of each species present in the system at each given time. Consider the case of the self-assembling robotic system. The CRN species correspond to the assembly configurations induced by the modules' embedded ruleset controllers. The CRN state is given by the vector $\vec{X}^{\text{macro}}(t) \in \mathbb{N}_{\geq 0}^{N_s}$ at each point in time, where the vector elements specify the number of individuals of each species. A reaction R may occur provided that the number of reactants is sufficient, that is, $\vec{X}^{\text{macro}} \geq \vec{r}_R$ element-wise. When reaction R occurs, the new state $\vec{X}^{macro}_{new}$ is given by:

$$\vec{X}^{macro}_{new} = \vec{X}^{macro} - \vec{r}_R + \vec{p}_R \tag{5}$$

A characterizing quantity for a reaction R is its propensity function a_R, defined such that $a_R(\overrightarrow{x}, .)dt$ is the probability that one instance of reaction R will occur in the next time interval $[t, t + dt)$ as $dt \to 0$, assuming the current state of the system to be $\overrightarrow{X}(t) = \overrightarrow{x}$. When the propensity function a_R is determined only by the current state of the system the Markov property holds and the time t until the next firing of reaction R can be described by an exponential random variable with mean $1/a_R(\overrightarrow{x})$, that is, its probability density is given by:

$$f(t) = a_R(\overrightarrow{x})e^{-a_R(\overrightarrow{x})t} \tag{6}$$

where $\overrightarrow{x}$ is the state of the CRN (i.e., a population vector), and $a_R(.)$ is the propensity function of the reaction R. The specific form of $a_R(.)$, assuming that the system is in dynamic equilibrium, is determined by the law of mass-action [19], and can be thus expressed as below:

$$a_R(\overrightarrow{x}) = k_R \tilde{a}_R(\overrightarrow{x}) \tag{7}$$

where k_R is the rate of reaction R and $\tilde{a}_R(\overrightarrow{x})$ has the appropriate form according to the stoichiometry of R, and does not depend on k_R. As an example, for the reaction $R : 1 + 2 \to 3$ the propensity function is computed with $\tilde{a}_R(\overrightarrow{x}) = N_1.N_2$ where N_1 and N_2 denote the number of reactants of type 1 and 2 in the system.

IV. Self-Assembly in our Robotic System

A. Procedure in the Real Setup

Our system consists of two main components: 1) the floating Lily robotic modules [20], and 2) the experimental setup built around them [7]. The Lilies are endowed with four custom-designed Electro-Permanent Magnets (EPM) to latch and also to communicate locally with their neighbors. Given a target structure, an appropriate ruleset is derived as detailed in Section IV-B, and deployed on all robots through wireless bootloading. The robots' EPM latches are by default enabled, resulting in a default latching upon meeting another robot. Once latched, the EPM-to-EPM inductive communication channel is physically established. The robots then exchange their internal states and look for an applicable rule in their ruleset. If no applicable rule is found, they unlatch by switching off their EPM latches; otherwise they remain latched and update their internal states accordingly. The Lilies are not self-locomoted, they are instead stirred by the flow field produced within a tank by several peripheral pumps. The evolution of the system is monitored through visual tracking of passive markers on top of the modules.

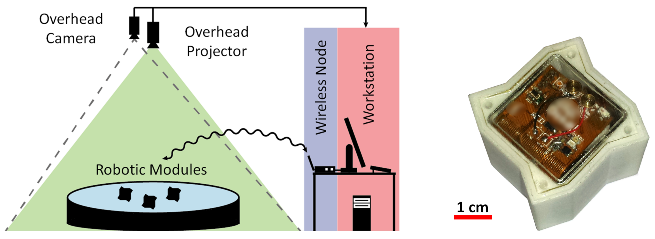

Fig. 1. An overview of the system: (left) The experimental setup: composed of water-filled tank with peripheral pumps agitating the fluidic environment, overhead camera and projector, wireless node for establishing radio link between the workstation and the robots; (right) The Lily robotic module.

B. Dynamics of Programmable Self-assembly

Here we describe a general mathematical model of programmable self-assembly of robotic modules based on the extended graph grammars formalism [21]. The state of our programmable self-assembling robotic system is described by an extended labeled graph over a fixed set of extended vertices representing the modules, and a dynamic set of edges representing the forming bonds.

Definition: An *extended vertex* has ordered link slots which correspond to the latching connectors of a robotic module, indexed following a counter-clockwise (CCW) rotation convention. We assume rotationally symmetric modules; for an isolated module the connectors are thus anonymous.

Definition: An *extended label* is a pair $l = (l_a, l_n)$ encoding the internal state of a module. l_a represents the control state of the robotic module and l_n represents the index of the most recently engaged connector.

Definition: An *extended labeled graph* is a quadruple $G = (V, E, S, \ell)$ where $V = \{1, ..., M\}$ is the set of extended vertices, $E \subset V \times V$ is the set of edges, $K = \{1, ..., N\}$ is the set of link slots available on each of the extended vertices, $S : E \to K \times K$ defines which slots are involved in a link between two vertices, and $\ell : V \to \Sigma$ is a labeling function, with Σ being a set of extended labels.

We employ a dedicated software framework which allows for automatically synthesizing rulesets directly pro-

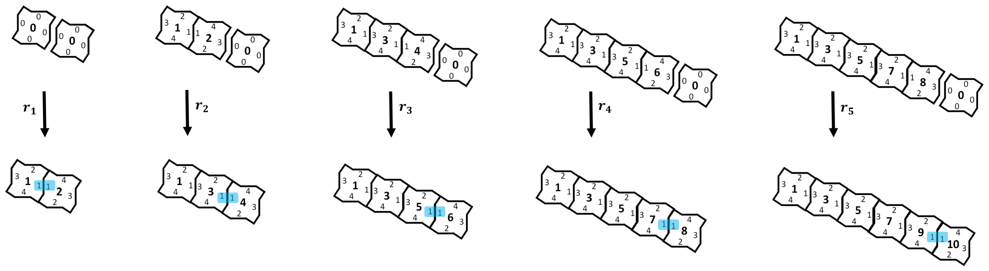

Fig. 2. Formation of a chain structure employing ϕ_{chain} on six Lily robotic modules. The most recently engaged connectors are marked in blue.

grammable on robotic modules [21]. The synthesized rules describe interacting modules' states using a combination of a control state variable and a relative hop number. The idea is that the robotic modules can only take part in a reaction defined by a certain rule if they have the right control state and are participating in the reaction with the appropriate orientation. Once a latching connector is engaged, the robot communicates its internal state in the form of a relative extended label of $l = (l_a, l_h)$ with l_a being the robot's control state and l_h being a relative hop number which represents the relative orientation of the currently engaged connector with respect to its predecessor, assuming a CCW hop convention. For a module with an internal state of (l_a, l_n) and N connectors, $l_h = [(l_n - l_c) \ mod \ N] + 1$, where l_n and l_c are the indexes of the most recently and the currently engaged connectors, respectively. For an isolated module the connectors are anonymous in terms of interaction possibilities, thus $l_h = 0$. In this work, we consider a case study on a chain shape target structure. Assuming that robotic modules are all initialized with a state of $(0, 0)$, the ruleset below allows for formation of a chain shape as depicted in Fig. 2. The forward rules, r_1 to r_5, advance structure formation, while the reverse rules, $\bar{r}_1$ to $\bar{r}_5$, allow for avoiding deadlocks [14] when executed probabilistically.

$$\phi_{chain} = \begin{cases} (0,0) & (0,0) & \leftrightharpoons & (1,1)-(2,1) & (r_1, \bar{r}_1) \\ (0,0) & (2,3) & \leftrightharpoons & (4,1)-(3,1) & (r_1, \bar{r}_1) \\ (0,0) & (4,3) & \leftrightharpoons & (6,1)-(5,1) & (r_1, \bar{r}_1) \\ (0,0) & (6,3) & \leftrightharpoons & (8,1)-(7,1) & (r_1, \bar{r}_1) \\ (0,0) & (8,3) & \leftrightharpoons & (10,1)-(9,1) & (r_1, \bar{r}_1) \end{cases}$$

For further simplification, we represent the simplified ruleset using the type of the configuration the robotic modules form (i.e. the structure size), as below:

$$\phi_{chain} = \begin{cases} 1+1 & \leftrightharpoons & 2 & (r_1, \bar{r}_1) \\ 1+2 & \leftrightharpoons & 3 & (r_2, \bar{r}_2) \\ 1+3 & \leftrightharpoons & 4 & (r_3, \bar{r}_3) \\ 1+4 & \leftrightharpoons & 5 & (r_4, \bar{r}_4) \\ 1+5 & \leftrightharpoons & 6 & (r_5, \bar{r}_5) \end{cases}$$

V. Simulation Methodology

Figure 3 depicts the submicroscopic simulation of the system. In this context, submicroscopic reflects the fact that the model provides a higher level of detail than a canonical microscopic model, faithfully reproducing intra-robot features (e.g., body shape, individual sensors and actuators). With this level of details, a submicroscopic simulator can keep track of a number of state variables such as the exact

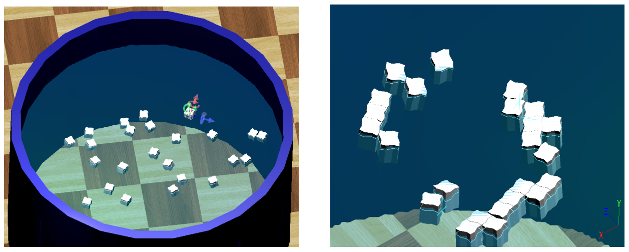

Fig. 3. The submicroscopic simulation of Lily robots in Webots [22].

pose of the robotic node, the specific forces exerted by one of its actuators, or the signal perceived by one of its sensors. In order to faithfully recreate our self-assembling system in simulation, we use Webots [22], a physics-based robotics simulator. Webots uses the Open Dynamics Engine (ODE) for simulating rigid body dynamics. Additionally, in order to simulate specific non-natively supported physics, it is possible to employ custom-designed physics plugins. The Lily robotic modules' CAD design as well as the robots' controller software are imported into the Webots simulated world. The rulesets programmed on the simulated robots are also identical to the case of the microscopic simulation in the previous section. The latest version of Webots supports a basic fluid node which allows for a simple uniform stream velocity, but is not capable of simulating a complex fluidic field. We used a similar approach as [23] to reproduce the complex flow field and the corresponding hydrodynamic forces. In particular, we developed a dedicated physics plugin for the simulated world in Webots that applies the drag force to the simulated Lily robotic module based on the velocity of the module and the flow velocity at its location at each time instant. The details of this development can be found in [24].

VI. Developing Markov Models

We use the CRN formalism as detailed in Section III-C to express a Markov model of our system and estimate the reaction rate constants of the network. The structure of the CRN model is fixed, with the species determined by the assembly configurations built by ϕ_{chain}, the parameters are estimated through different methods. It can be shown that for a given set of observations of the system, i.e. a sequence of events $(e_1, ..., e_n)$, with $e_i = (R_i, t_i, \overrightarrow{x}_i)$, the Maximum Likelihood (ML) estimator of the rate vector

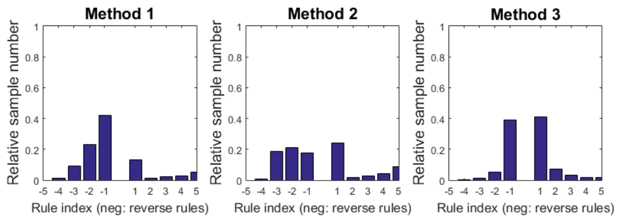

Fig. 4. Normalized number of samples per rule gathered through the three methods of Section VI.

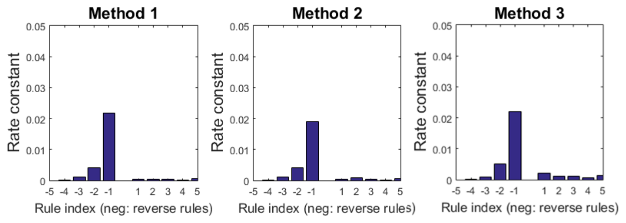

Fig. 5. Estimated rate constants using the three methods of Section VI.

$\overrightarrow{k} = [\hat{k}_1, ..., \hat{k}_{N_R}]$ of the underlying CRN is [6]:

$$\hat{k}_j = \frac{\Sigma_{i=1}^{n} \mathbf{1}_{R_i = R_j}}{\Sigma_{i=1}^{n} (t_i . \tilde{a}_{R_j}(\overrightarrow{x}_i)}) \qquad j = 1, ..., N_R \tag{8}$$

Given the high-fidelity simulation framework described in Section V, one can evolve the system dynamics from any desired initial condition and for any desired duration of time. The essential idea for estimating the rate constants is to observe and record the different reaction events and the corresponding waiting times in between.

Different methods have been proposed for gathering the statistics necessary for the rate estimation. We exploit the high-fidelity submicroscopic simulation framework described in Section V and study two existing methods and propose a third new method for gathering statistics. The main difference between these methods is the conditions under which they sample an event occurence and also the statistical composition of the dataset they gather. Methods 1 and 2 initialize and reinitialize the system to various macrostates upon occurrence of different events. Method 3, on the other hand, gathers observations from a full-length simulation of the system, initializing the simulation to fully isolated modules for each set of observations.

Method 1: This method has been originally proposed in [1]. The authors propose to run simulations for a wide variety of initial macrostates, covering all the states traversed by the system during the SA process. Formally, the initial conditions are defined as follows:

$$s_M(t = 0) = (S_1, S_2, \ldots, S_n)$$

for all feasible macrostates in the system. The simulation is initialized and run for each initial macrostate distribution, as soon as a reaction R occurs, the time of that reaction is noted and the simulation is reset.

Method 2: This method has been originally proposed in [5] and is similar to the Method 1 in that it also considers only a simulation period until a reaction occurs in the system. However, the number of different initial conditions is reduced. For computing the reaction rate for a specific reaction R, the initial macrostates used in the simulations are only the ones containing the reactants of reaction R. For instance, given the stochastic reaction equation $1 + 3 \rightarrow 4$, which describes the formation of substructure 4 from substructures 1 and 3. This would consequently require the following corresponding initial conditions:

$$s_M(t = 0) = (S_1, 0, S_3, 0) \quad , \quad S_i = 0 \quad \forall i \neq \{1, 3\}$$

The values of the number of copies of substructures S_i are randomly varied within the feasible macrostates. The simulation is thus initialized and run for each initial macrostate distribution, as soon as reaction R occurs, the time of that particular reaction is noted and the simulation is reset but the time is not reset.

Method 3: This is our proposed method. The idea is simply that observing entire simulations starting from different initial conditions (all the agents separated at start) until the target structures are ultimately built. This should provide a wide variety of interactions between the agents distributed according to the natural tendency of the system, allowing therefore to collect enough relevant statistical information to determine rate constants. All initial conditions are defined as follows:

$$s_M(t = 0) = (N_0, 0, \ldots, 0)$$

where the first element in the macrostate vector indicates the species corresponding to an isolated module, thus, $S_1 = N_0$.

For gathering the statistics, our simulation setup consists of 12 robotic modules that are stirred in the fluidic arena of 1.2 m diameter. The evaluation was performed for the ϕ_{chain} ruleset that builds the target structure of a chain using six robotic modules as described in Section IV-B. All the forward rules are set to be executed with probability 1, for the reverse rules the probabilities were set to the following values 0.05, 0.01, 0.002, 0.0004, 0 for $\{\bar{r}_1, \bar{r}_2, \bar{r}_3, \bar{r}_4, \bar{r}_5\}$ respectively. Figure 4 shows the statistics of the number of sample points gathered while the estimated values of the rates are shown in Figure 5. It can be seen that the statistical composition of the datasets gathered by the three methods are very different. In addition to the different number of samples per reaction rule, the system state at which the reaction time has been sampled is also different for the three methods, resulting in different rate estimates. While the differences between the estimated rates as depicted in Figure 5 seem minor, they describe substantially different system evolution courses as depicted in Figure 6. We generate 1000 sample trajectories using the Gillespie method [19] with the Stochkit software for each CRN model. It is also noteworthy that Method 1 provides the most varied system state at the sampling time, while Method 3 samples the reaction events at system states through which the system has a natural tendency to traverse.

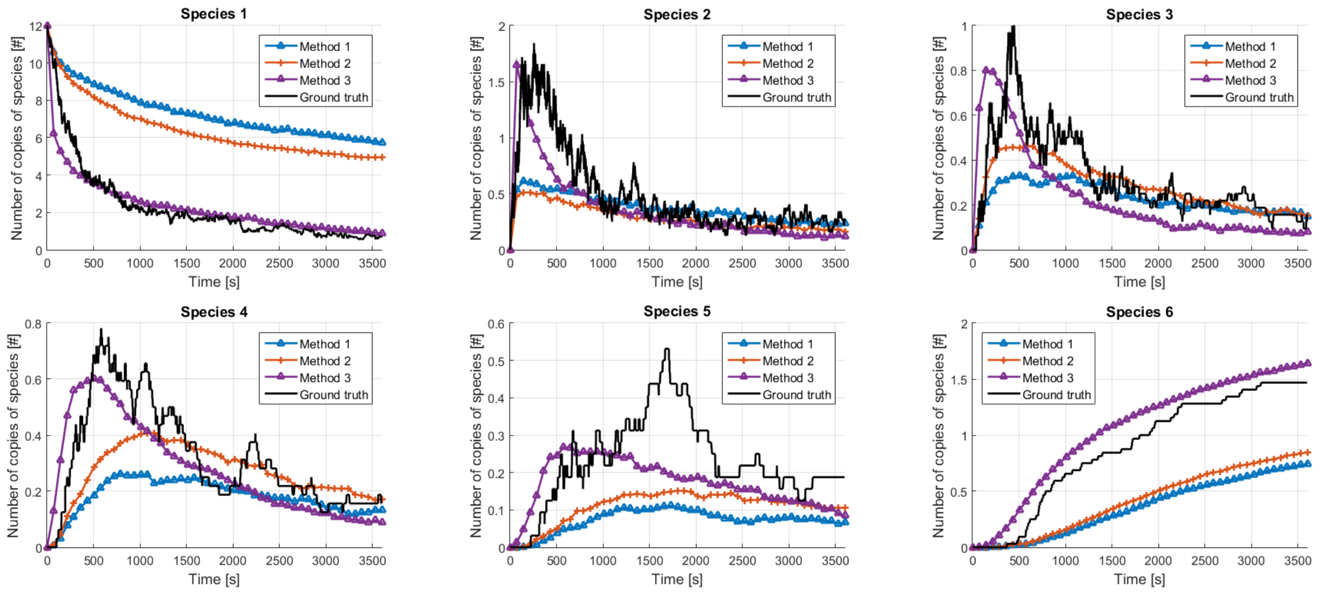

Fig. 6. Comparison of model prediction regarding the trajectory of each of the six species, averaged 1000 runs in Stochkit, formed by the ϕ_{chain} using the rates estimated by the three methods in Section VI. The ground truth is obtained by averaging 90 runs of the simulated world in Webots (see Section V).

VII. Evaluating Well-Mixed Condition

The mismatch between the Markov model predictions and the ground truth as depicted in Figure 6 can be ascribed to two factors. First, the underlying CRN describing the states of the Markov model has its species determined by the ruleset. However, as a result of random interactions between the robotic modules, temporary bindings can also form, which are later severed as the modules switch their EPMs off and drift apart. These temporary bindings/structures are not explicitly modeled as the CRN species. Second, the physical dynamics of the system might be practically far from the ideal well-mixed conditions, thus voiding the Markov property assumption. In this section, we specifically look into validation of the well-mixed condition. The well-mixed condition is sufficient for guaranteeing that the underlying process is Markovian and that each possible combination of reactants for a particular reaction will be equi-probably involved in the next instance of the reaction [5].

We use the definition of well-mixedness from [19]: The rate at which new collisions occur should be greater than the rate at which reactions occur [19], in other words the modules should diffuse faster than they react. Similar to the work in [5], we rely on diffusion for module transportation. A measure of "well-mixed" as proposed in [5] is to require that $D/k_{av} > A$ where D is the diffusion coefficient, k_{av} is a nominal reaction rate in the system and A is the effective area of the fluidic arena occupied by the interacting modules. We study two methods for estimating the diffusion coefficient, one proposed in the previous literature [5] and a new method. In order to gather statistics, we exploit again our high-fidelity simulation framework described in Section V and conduct two simulated experiments with 12 robotic modules each for a duration of 10 min. The robotic modules are programmed with an empty ruleset controller and as a result unlatch right after they latch.

A. Method 1

This method has been originally proposed in [5]. The authors propose to estimate the diffusion coefficient for a robotic module as below:

$$D = \frac{E(r^2(t))}{4t} \tag{9}$$

where $r(t)$ denotes the random displacement of the robot as a function of time. We compute the expected value considering all the robotic modules and all the time steps. The final obtained estimated diffusion coefficient using this method is 0.0016 m^2/s. For the nominal reaction rate we consider the order of magnitude of the rate of the rule r_1 in the ruleset ϕ_{chain} as computed by Method 3, 0.001. The condition of $D/k_{av} > A$ implies that the arena size should be smaller than $A_{max} = 1.6m^2$ while the area of our arena is $1.13m^2$. We thus conclude that the system is well-mixed.

B. Method 2

Fick's law of diffusion gives a relation between the diffusion flux, the gradient in concentration and a diffusion coefficient under the assumption of the system being in steady state. It can be expressed as:

$$J = -D\nabla_c \tag{10}$$

Where in the case of a 2D diffusion, J is the rate of robot transfer per boundary length normal to the direction of transfer, expressed in $[robots/m.s]$, D is the diffusion coefficient, expressed in m^2/s, and ∇_c is the gradient in concentration, expressed in $[robots/m^3]$. In order to estimate the diffusion coefficient in the case of our 2D system, the arena is partitioned by a square grid of 16x16 with each cell i

containing a concentration of robots c_i (where $c_i \in \mathbb{N}^{+0}$). For each simulation time step Δt, the modules' flow is computed across each cell boundary (e.g. from cell i to cell j) of a 4-connected neighborhood, using a discrete expression of the Fick's law:

$$F_{i \to j} = -D(c_j - c_i)\Delta t \tag{11}$$

To estimate the diffusion coefficient, we write:

$$D_{c_i \to c_j} = -\frac{F_{c_i \to c_j}}{(c_j - c_i)\Delta t_{c_i \to c_j}} \tag{12}$$

The diffusion coefficient is thus computed for every pair of observed concentrations (c_i, c_j). The result of this diffusion coefficient estimation is a 2D matrix. To investigate the well-mixed condition, individual values are then averaged to have a single value for the estimated D. The final estimation for the diffusion coefficient using this method is 0.00065 m^2/s. Referring to the well-mixed condition described as $D/k_{av} > A$, we should have an $A_{max} = 0.65m^2$. While the total area of the arena is $1.13m^2$, the effective area to which the motion of the modules is limited is smaller, about half of the total available arena size. We thus conclude that the system is border-line well-mixed.

VIII. Developing Hidden Markov Models

The goal here is to construct a CRN corresponding to an HMM. Based on an existing CRN $\mathcal{N}$ model of the system, we provide a method to automatically obtain a refined CRN $\widetilde{\mathcal{N}}$ model, where the original species are augmented with proper hidden species. We take the systematic approach in [6] and employ a metric in order to provide for a fully automatic refinement method. The original approach starts from a given CRN $\mathcal{N} = (\mathcal{R}, \mathcal{S})$ and an associated set of events $(e_1, \ldots, e_n)$, and attempts to construct a refinement $\widetilde{\mathcal{N}} = (\widetilde{\mathcal{R}}, \widetilde{\mathcal{S}})$ of $\mathcal{N}$ such that the likelihood $\mathcal{L}(\widetilde{\vec{k}}|e_1, \ldots, e_n) > \mathcal{L}(\vec{k}|e_1, \ldots, e_n)$, i.e., the sequence of events is better explained by the refined model than the original one. In order to refine a CRN, a species S_0 is selected. It is then split into two subspecies S_0^a and S_0^b. All reactions involving S_0 are then duplicated, with their corresponding reactants and products updated accordingly. Consider R_j:

$$R_j: \quad S_0 + 2\,A_1 \rightharpoonup 3\,A_2 \tag{13}$$

it will be duplicated as below:

$$R_j^a: \quad S_{0,a} + 2\,A_1 \rightharpoonup 3\,A_2 \tag{14}$$

$$R_j^b: \quad S_{0,b} + 2\,A_1 \rightharpoonup 3\,A_2. \tag{15}$$

In contrast to the approach proposed in [10], the resulting subspecies $S_{0,a}$ and $S_{0,b}$ are not specifically associated to the type of a previous interaction partner. More specifically, if an event $e_i = (R_i, t_i, \vec{x}_i)$ is such that R_i is duplicated, one needs to re-assign this event to either $R_i = R_i^a$ or $R_i = R_i^b$, and update accordingly the population vectors $\vec{x}_{j>i}$ of the upcoming events. To solve this problem, an expectation-maximization (EM) algorithm is used, an iterative method for finding maximum likelihood estimates of parameters in statistical models that depend on unobserved latent variables [6]. We introduce modifications in the original method in two ways: first, in order to automatically choose the species to be refined we propose a metric based on evidence from the observed set of events as below:

$$h = \frac{\sigma(\Delta t_i)}{m(\Delta t_i)} \tag{16}$$

Where for a reaction of interest, $\sigma(\Delta t_i)$ is the standard deviation of all the observed waiting times and $m(\Delta t_i)$ is the median of the observed waiting times. h is computed for all species in the CRN and the species with the maximum value of h will be refined next. Additionally, we introduce a stopping condition for the refinement procedure based on the same metric: the refinement process would thus be halted when the maximum h in the CRN is less than a predefined threshold t_h, set to 0.3. In order to keep the size of the CRN model tractable, here we perform one round of refinements starting from the CRN corresponding to the best model in Section VI, i.e. Method 3. In order to generate trajectories for the refined CRN, the original initial condition of fully isolated modules should be also refined to consider the initialization of the hidden species as well. Using the Stochkit software, we generate 1000 sample trajectories for each initial condition of the refined CRN corresponding to the HMM. This is a crucial part in evaluating the HMM accuracy which has not been addressed in previous works. The average trajectory of the HMM is then compared with the ground truth provided by high-fidelity simulations of the system as detailed in Section V. The results are depicted in Figure 7. It can be seen that the model accuracy is significantly improved and the average trajectories match the ones of ground truth very well. Species 5 exhibits an interesting characteristics, the sample trajectories corresponding to different initial states differ largely compared to those of the other species, and the HMM does not manage to capture the ground truth trajectory peak. We speculate that a deeper refinement corresponding to an HMM with more hidden species in combination with a larger dataset should achieve better accuracy.

IX. Conclusion and Future Work

In this work, we considered the case of our floating self-assembling robotic system and investigated developing macroscopic probabilistic models of programmable stochastic self-assembly of our system through multiple methods. A high-fidelity simulation of the system was used as the ground truth. We utilized the CRN formalism to express the SA mechanism in the system. The contributions of this work are along three axes (1) a new rate estimation method is introduced and compared with two existing ones, (2) a new method for estimating diffusion coefficient is studied, and (3) an automatic method for creating an HMM starting from a Markov model is formulated based on a previously existing systematic method. We show that assuming the well-mixed assumption and starting from different Markov models of the system, the hidden states augmented through our automatic

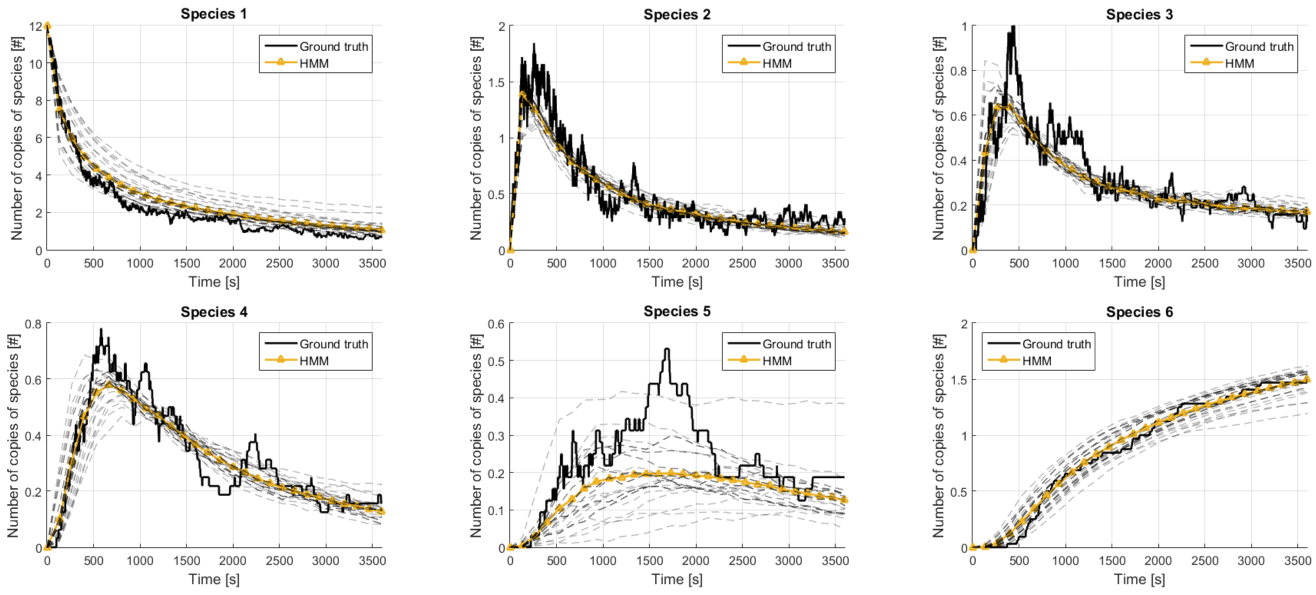

Fig. 7. Comparison of model prediction regarding the trajectory of each of the six original species formed by the ϕ_{chain} using the rates estimated by the Method 3 in Section VI after the CRNs have been refined. The gray dashed curves are sample trajectories corresponding to different initial HMM states.

HMM refinement method improve the model prediction accuracy, compensating the imprecise model assumptions. In the future, we plan to formalize and consolidate the approach investigated in this work and propose a methodology for developing Markovian models for general programmable self-assembling systems.

REFERENCES

[1] V. Ganesan and M. Chitre, "On stochastic self-assembly of underwater robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 251–258, 2016.

[2] B. Haghighat, M. Mastrangeli, G. Mermoud, F. Schill, and A. Martinoli, "Fluid-mediated stochastic self-assembly at centimetric and sub-millimetric scales: Design, modeling, and control," *Micromachines*, vol. 7, no. 8, p. 138, 2016.

[3] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[4] E. Tuci, R. Groß, V. Trianni, F. Mondada, M. Bonani, and M. Dorigo, "Cooperation through self-assembly in multi-robot systems," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 1, no. 2, pp. 115–150, 2006.

[5] N. Napp, S. Burden, and E. Klavins, "The statistical dynamics of programmed self-assembly," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1469–1476, 2006.

[6] G. Mermoud, *Stochastic Reactive Distributed Robotic Systems*. Springer, 2014.

[7] B. Haghighat and A. Martinoli, "Characterization and validation of a novel robotic system for fluid-mediated programmable stochastic self-assembly," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2778–2783, 2016.

[8] L. Matthey, S. Berman, and V. Kumar, "Stochastic strategies for a swarm robotic assembly system," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1953–1958, 2009.

[9] T. P. Pavlic, S. Wilson, G. P. Kumar, and S. Berman, "Control of stochastic boundary coverage by multirobot systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 3, p. 034504, 2015.

[10] N. Napp, D. Thorsley, and E. Klavins, "Hidden markov models for non-well-mixed reaction networks," in *2009 American Control Conference*, pp. 737–744, 2009.

[11] T. W. Mather and M. Ani Hsieh, "Macroscopic modeling of stochastic deployment policies with time delays for robot ensembles," *International Journal of Robotics Research*, vol. 30, no. 5, pp. 590–600, 2011.

[12] N. Correll and H. Hamann, "Probabilistic modeling of swarming systems," in *Springer Handbook of Computational Intelligence*, pp. 1423–1432, 2015.

[13] M. Tolley and H. Lipson, "Programmable 3d stochastic fluidic assembly of cm-scale modules," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4366–4371, 2011.

[14] E. Klavins, "Programmable self-assembly," *IEEE Control Systems*, vol. 27, no. 4, pp. 43–56, 2007.

[15] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[16] S. C. Goldstein, J. D. Campbell, and T. C. Mowry, "Programmable matter," *Computer*, vol. 38, no. 6, pp. 99–101, 2005.

[17] W.-M. Shen, P. Will, and B. Khoshnevis, "Self-assembly in space via self-reconfigurable robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 2516–2521, 2003.

[18] H. Yan, S. H. Park, G. Finkelstein, J. H. Reif, and T. H. LaBean, "DNA-templated self-assembly of protein arrays and highly conductive nanowires," *Science*, vol. 301, no. 5641, pp. 1882–1884, 2003.

[19] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The journal of physical chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[20] B. Haghighat, E. Droz, and A. Martinoli, "Lily: A miniature floating robotic platform for programmable stochastic self-assembly," in *IEEE International Conference on Robotics and Automation*, pp. 1941–1948, 2015.

[21] B. Haghighat, B. Platerrier, L. Waegeli, and A. Martinoli, "Synthesizing rulesets for programmable robotic self-assembly: A case study using floating miniaturized robots," in *International Conference on Swarm Intelligence*, pp. 197–209, Springer, 2016.

[22] O. Michel, "Webots: Professional mobile robot simulation," *Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[23] E. Di Mario, G. Mermoud, M. Mastrangeli, and A. Martinoli, "A trajectory-based calibration method for stochastic motion models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4341–4347, 2011.

[24] B. Haghighat and A. Martinoli, "A rule synthesis algorithm for programmable stochastic self-assembly of robotic modules," in *To appear in proceedings of the International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2016.