2017 American Control Conference
Sheraton Seattle Hotel
May 24–26, 2017, Seattle, USA

Semi-supervised Pattern Synthesis in Spatially Distributed Dynamical Systems

Iman Haghighi, Kevin Leahy, Rachael Ivison, and Calin Belta

Abstract— **Pattern producing networks of dynamical systems are useful in a variety of applications ranging from formation control in multi-agent robotics to biological networks. We propose a formal methods framework with minimal user input to synthesize system parameters that result in the emergence of global steady state patterns in spatially distributed dynamical systems. Our framework consists of extensive exploration of steady state behaviors, dividing these behaviors into groups of patterns using a clustering algorithm, and determining system parameters for each pattern cluster using a recently developed formal methods approach for parameter synthesis. A case study illustrating the implementation of this framework on a network of locally interacting living cells is included.**

I. INTRODUCTION

Understanding and controlling the capabilities of networks of interacting systems is a challenging problem, with many important applications [1]. It is well known that the interactions of simple, identical dynamical systems can produce a variety of interesting behaviors and properties. The production of such behavior is known as emergence [2]. We aim to understand how local interactions in a network can be tuned so that the entire network produces a specific emergent behavior. The answer to this question is useful in numerous areas such as multi-agent robotics [3] and cellular automata [4]. We frame the question of such emergent behaviors through the lens of formal pattern recognition and synthesis.

Pattern synthesis has become a topic of particular interest in recent years due to developments in the area of engineering living cells. Synthesizing emergent behaviors in such systems is a first step to solving more fundamental problems such as embryogenesis [5], and organoid formation [6]. Current approaches to this problem are usually application-specific [7], [8], and the existence of a formal framework that does not make narrow assumptions on the properties of the network and is applicable to a wide variety of applications is an open question.

Pattern recognition is normally formulated as a supervised machine learning problem [9]. A central assumption in this approach is the existence of a labeled dataset containing desirable and undesirable patterns. This assumption may not be practical for applications in which acquiring a large enough training set through experiments is infeasible and/or the range of patterns that can emerge is not known a priori [10]. Hence, we aim to develop a semi-supervised framework that is able to synthesize system parameters that produce patterns without the need to specify those patterns beforehand.

There has been an increasing effort in recent years to use formal logics as descriptors of spatial properties. Linear spatial superposition logic (LSSL) was successfully used in [11] to identify self-similar texture. This paper is closely related to [12], in which the authors propose a formal methods approach to synthesize Turing patterns [13] in a network of locally interacting cells. They use a supervised learning procedure to learn formal pattern descriptors expressed as a formula in a logic called tree spatial superposition logic (TSSL) given a labeled dataset of images containing desirable and undesirable patterns. In this paper, we modify this algorithm so that a labeled dataset is no longer needed. We start by exploring the state space for different possible steady state behaviors, cluster those behaviors into distinct patterns using a measure of image similarities [14], and use the algorithm in [12] to synthesize system parameters for pattern cluster(s) that an expert user identifies as desirable.

Numerous image similarity measures have been proposed in recent years for a wide variety of applications. [15] presents a word co-occurrence model specifically suitable for person re-identification. An online ranking algorithm for computing image similarities is developed in [16] that is capable of studying large-scale systems. In this paper, we use a similarity distance measure proposed in [14] based on Kolmogorov complexity [17]. This metric was effective for clustering Turing patterns in biological systems.

II. PROBLEM STATEMENT

Consider a set of $M \times N$ identical interacting dynamical systems distributed on a grid. We denote the system located on the m^{th} row and n^{th} column of the grid as $S_{m,n}$. The collection of all the systems is denoted by $\mathbb{S}$. The state of $S_{m,n}$ is written as $x_{m,n} \in \mathcal{X} \subseteq \mathbb{R}^{N_x}$ and evolves according to the differential equation

$$\dot{x}_{m,n} = f(x_{m,n}, u_{m,n}, p), \tag{1}$$

where $u_{m,n} \in \mathbb{R}^{N_u}$ is a set of inputs, $p \in \Omega \subseteq \mathbb{R}^{N_p}$ is a set of parameters, and $f : \mathcal{X} \times \mathbb{R}^{N_u} \times \Omega \to \mathbb{R}^{N_x}$ describes the dynamics of the system. The parameters p are constant for a given system. The global behaviors of the network can be controlled by choosing proper values for p. A set of observations $y_{m,n} \in \mathbb{R}^{N_y}$ for each system is defined by

$$y_{m,n} = h(x_{m,n}), \tag{2}$$

The authors are with Boston University, Boston, MA 02215, USA {haghighi,kjleahy,rivison,cbelta}@bu.edu

This work was partially supported at Boston University by the NSF under grants CNS-1446607, CBET-0939511, and by the ONR under grant MURI 014-001-0303-5.

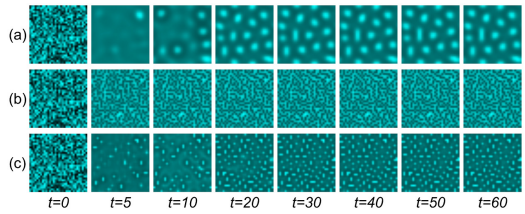

Fig. 1. Pattern emergence in the Turing system (3) with three different parameterizations (a) Large spots (b) Fine patches (c) Small spots.

where $h : \mathcal{X} \to \mathbb{R}^{N_y}$ is an output (observation) function.

While the output of an isolated system may be simple to analyze, the interaction of $M \times N$ such systems may be infeasible to predict analytically. We consider the case in which a user does not have a priori knowledge of the behavior the systems exhibit via interaction with each other. If such a user wishes to analyze the systems' behavior, it may be necessary to empirically test the system. In this paper, we consider such a case, focusing on systems that have reached a steady state. We will henceforth call the spatial configuration of steady state observations a *pattern* if that configuration is of interest to an expert user based on the specific application.

Example 1: Consider a network of reaction-diffusion systems that describe the generation of skin pigments that produce spots in animals [13]. We consider a 32×32 reaction-diffusion system with two species. The concentrations of the species in $S_{m,n}$ evolve according to the ODEs

$$\begin{aligned} \frac{dx^{(1)}_{m,n}}{dt} &= D_1(u^{(1)}_{m,n} - x^{(1)}_{m,n}) + R_1 x^{(1)}_{m,n} x^{(2)}_{m,n} - x^{(1)}_{m,n} + R_2, \\ \frac{dx^{(2)}_{m,n}}{dt} &= D_2(u^{(2)}_{m,n} - x^{(2)}_{m,n}) + R_3 x^{(1)}_{m,n} x^{(2)}_{m,n} + R_4, \end{aligned} \tag{3}$$

where $x^{(1)}_{m,n}$ and $x^{(2)}_{m,n}$ represent the concentration of each species at cell (m, n), $u^{(1)}_{m,n}$ and $u^{(2)}_{m,n}$ are the inputs of the (m, n)th system $S_{m,n}$ from neighboring systems,

$$u^{(k)}_{m,n} = \frac{1}{|\gamma_{m,n}|} \sum_{\gamma \in \gamma_{m,n}} x^{(k)}_{\gamma}, \tag{4}$$

where $\gamma_{m,n}$ is the set of indices of systems adjacent to $S_{m,n}$, $D_i, i \in \{1, 2\}$ are diffusion coefficients, and $R_i, i \in \{1, \ldots, 4\}$ are the reaction parameters that define the local dynamics for the two species. The diffusion and reaction rates are the set of parameters that can control pattern emergence in (3), so

$$p = \langle D_1, D_2, R_1, R_2, R_3, R_4 \rangle. \tag{5}$$

The normalized concentrations of the second species are observable. Formally,

$$y_{m,n} = \frac{x^{(2)}_{m,n}}{\max\limits_{m \le M, n \le N} x^{(2)}_{m,n}}. \tag{6}$$

Fig. 1 shows the observed concentrations of species 2 at different time points for three different parameter choices. The brighter the cell (m, n) is, the larger the relative value of $x^{(2)}_{m,n}$. The three corresponding steady-state spatial patterns, called large spots (LS), fine patches (FP), and small spots (SS), can be clearly seen at time $t = 60$ (steady state). Our goal in this work is to automatically classify these types of behaviors with minimal user input. Further, we wish to identify the parameters that result in the production of such patterns with high probability.

For a system such as that described in Example 1, we define the problem more formally as follows.

Problem 1: Given a network of identical dynamical systems distributed on a grid $\mathbb{S}$ (1), a range of initial conditions $\mathcal{X} \subseteq \mathbb{R}^{N_x}$, and a parameter search space $\Omega \subseteq \mathbb{R}^{N_p}$:

1) Explore the state space for groups of steady state behaviors that are found to be interesting patterns by an expert user.
2) If such patterns were found in the previous step, determine parameters $p^* \in \Omega$ such that the corresponding steady state observations are guaranteed to resemble the desired patterns.

Our approach to Problem 1 can be summarized as follows. First, we collect different possible steady state behaviors of (1) by extensively simulating it with random initial conditions and parameters. Next, we use image similarity measures and a clustering algorithm [14] that divides the steady state images produced by simulation into groups of similar patterns. At this point, an expert user identifies image clusters that are of particular interest to them, based on the application. In the next step, we use a formal methods approach presented in [12] to learn a descriptor for each cluster expressed in a formal language called tree spatial superposition logic (TSSL). TSSL is equipped with a metric that quantifies how far an image is from a pattern. This metric is used to formulate an optimization procedure to synthesize system parameters that maximize the emergence of patterns belonging to the clusters that were previously identified by the user. The details of the solution are presented in Sec. III.

III. SOLUTION

A. State Space Exploration

The purpose of this paper is to synthesize global patterns in networks of locally interacting dynamical systems without a priori knowledge of the patterns that the network is capable of producing. Consequently, we need to start by exhaustive simulation of (1) with parameter values randomly chosen from a uniform distribution in Ω and random initial conditions chosen from $\mathcal{X}$ in order to collect a large set of steady state behaviors. At the end of this step, the set images demonstrating different steady state behaviors is saved and denoted by $\mathcal{Y}$, with $|\mathcal{Y}|$ being the total number of images and $p^{(y)}$ the parameterization that led to image $y \in \mathcal{Y}$.

The problem of how many simulations are needed to cover all possible steady state behaviors falls under the general category of state space exploration [18] which is an open problem, application-specific, and out of the scope of this paper. Here, we assume the total number of images $|\mathcal{Y}|$ is pre-determined by an expert.

B. Similarity Measures and Clustering

Once the state space has been sampled, the task at hand is to determine similarity among the images in $\mathcal{Y}$ so that we can learn the parameter values that give rise to patterns of interest. Typically, an expert must label positive and negative samples for such a learning task, but we propose a semi-supervised learning method to reduce the workload. This is not only convenient for the expert, but also necessary from a technical standpoint, because we assume no a priori knowledge of the various types of system output. Since a sufficiently exhaustive exploration of the state space generates a large amount of data, categorizing the output manually is, at the very least, a difficult task. Instead, we use a *similarity distance measure* (SDM) from [14] to cluster images based on their similarity. The clustered data are presented to the expert, who may then decide if patterns of interest are present, and therefore determine the positive samples for TSSL inference (Sec. III-C).

The SDM is inspired by the idea of Kolmogorov complexity [17]. Essentially, two images are similar if one can be transformed into an approximation of the other without much manipulation. We present an overview of SDM here, but readers are directed to [14] for complete details.

Given an output image y_1 from a steady-state system, we learn a dictionary $\mathcal{D}_1$ and atoms $\mathbf{a}_1$ that approximate a solution to

$$y_1 = \mathcal{D}_1 \mathbf{a}_1 \,. \tag{7}$$

The image's *sparse complexity* $S(y_1, \mathcal{D}_1)$ is given by

$$S(y_1, \mathcal{D}_1) = \frac{1}{k} \sum_{i=1}^{k} ||\mathbf{a}_1|| \,, \tag{8}$$

where k is the dimension of $\mathbf{a}_1$. To compare image y_1 with a second image y_2, the authors compute the sparse complexity of each image with the dictionary learned for the other (i.e., $S(y_1, \mathcal{D}_2)$ and $S(y_2, \mathcal{D}_1)$). Then the SDM of the two images is given by

$$SDM(y_1, y_2) = \frac{S(y_1, \mathcal{D}_2) + S(y_2, \mathcal{D}_1)}{S(y_1, \mathcal{D}_1) + S(y_2, \mathcal{D}_2)} \,. \tag{9}$$

By computing SDM for all pairwise sets of images, we may cluster the data hierarchically, using standard techniques.

To cluster the data, we construct a dissimilarity matrix using the pairwise SDM for all images. This allows us to merge images into clusters according to a measure of *linkage strength*, which allows comparisons of clusters with more than one image, since SDM only compares two individual images. Once the clustering is complete, the data can be organized into a *dendrogram*, a tree-like graph that shows the organization of the clusters from a root with all the images to leaves containing only individual images (Fig. 2). Interested readers may find more information on clustering in many sources, such as [19]. Although this measure of image similarity fits our particular learning task, other techniques can be used depending on the problem being examined.

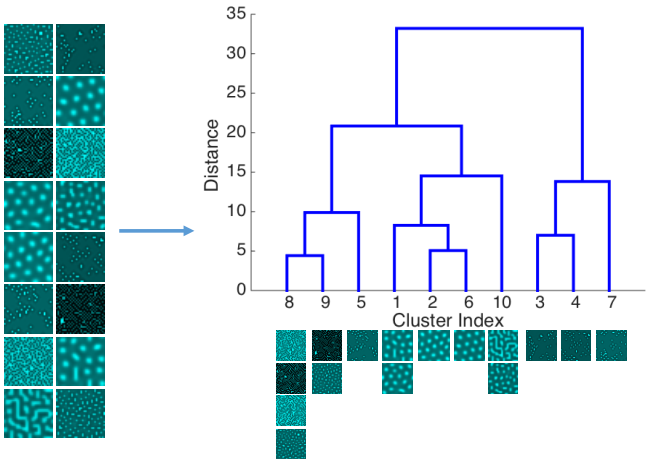

Fig. 2. An example from the system of Example 1 of clustering images into a dendrogram using SDM (9).

C. Parameter Synthesis

At this point, we have a dendrogram that divides steady state observations obtained from (1) (Sec. III-A) into groups of similar images (Sec. III-B). Next, an expert user needs to decide which clusters are interesting for the purpose of pattern synthesis. This will be achieved through a top-down approach. The root of the dendrogram corresponds to the set of all images that are being considered. Nodes located on lower level correspond to smaller sets of images since each vertex corresponds to a subset of its parent in the tree. The user will be consecutively presented with lower level clusters until one or more clusters are found in which most of the images demonstrate similar global patterns that would be of particular interest. These clusters of images are denoted by $Y_i,\ i = \{1, \cdots, \mathcal{C}\}$ where $\mathcal{C}$ is the total number of image clusters chosen by the expert. Notice that since all the images were obtained by computer simulations of (1) from constant parameter sets p and initial conditions, we already have a set of parameter sets from which it is possible to get steady state patterns belonging to one of the chosen image clusters. The set of parameters from which the images of cluster Y_i were obtained are denoted by

$$\mathcal{P}_i = \{p^{(j)} | j = 1, \cdots, |Y_i|\}, \tag{10}$$

where $|Y_i|$ is the cardinality of set Y_i and $p^{(j)}$ is the parameter set that was used to obtain the jth image in Y_i.

In this section, the objective is to synthesize system parameters p_i^* such that steady state behaviors of (1), when simulated with the parameter set p_i^*, "best" resemble the images belonging to Y_i. Due to stochasticities of the system (e.g., random initial conditions), it is not known how likely it is that one of the parameters in $\mathcal{P}_i$ will lead to a desired pattern. We intend to find a p_i^* with a high likelihood that steady state observations will be as close as possible to images in a desired cluster Y_i. The framework that we use to solve this problem was first introduced in [12]. The authors propose a formal methods approach in which global patterns are formally expressed as a logical formula

in a formal language called tree spatial superposition logic (TSSL). TSSL is equipped with quantitative semantics and can be used to construct a metric that can score how close or far an image is from given a pattern. We use this metric to formulate an unconstrained optimization problem. The solution to this problem provides a candidate for p_i^*. The reader can refer to [12] for a detailed explanation of this approach. Here, we present a brief informal explanation.

Consider the steady state observations of an $M \times N$ network as an $M \times N$ matrix y where the element $y_{m,n} = \langle y_{m,n}^{(1)}, \cdots, y_{m,n}^{(N_y)} \rangle$ is the set of observations for the system located on the mth row and nth column of the grid. Following [12], we represent the matrix y as a quad-tree. A quad-tree $Q = (\mathcal{V}, R)$ is a quaternary tree representation of matrix y where each vertex $v \in \mathcal{V}$ represents a submatrix of y and the relation $R \subset \mathcal{V} \times \mathcal{V}$ defines four children for each vertex that is not a leaf.

Example 2: Fig. 3 demonstrates how a quad-tree is built from a matrix through an example. We label each edge in the quad-tree with the direction of the submatrix represented by the child: NW, NE, SW, and SE. First, v_0 represents the entire matrix y. Second, $v_1 - v_4$ represent the four quadrants of v_0. Third, $v_5 - v_8$ represent the four quadrants of v_1. Since no quadrants after v_1 are further subdivided, we are done labeling. Next, we construct the quad-tree from the subdivided matrix, where each node's children are the corresponding vertices' subdivided quadrants. Each edge is labeled with the direction of the child's quadrant. A leaf is defined as a vertex of the quad-tree for which all the elements of a submatrix have the same values.

We define the representation function $\mu^{(c)}(v) : \mathcal{V} \to \mathbb{R}^2$ for a sub-matrix represented by vertex $v \in \mathcal{V}$ of the quad-tree $Q = (\mathcal{V}, R)$ as $\mu^{(c)}(v) = (\mu_1^{(c)}, \mu_2^{(c)})$ where μ_1 and μ_2 denote the first and second statistical moments of the submatrix, and $c \in \{1, \cdots, N_y\}$. Thus, the function $\mu^{(c)}$ provides the mean value and variance of the observations in a particular region of the space represented by the vertex v.

A TSSL formula is recursively constructed using the following:

- Linear predicates over values for the representation function. e.g., $\mu_1^{(1)} > \lambda_1$, $\mu_2^{(1)} < \lambda_2$.
- Boolean operators: e.g., $\neg\phi$, $\phi_1 \wedge \phi_2$, and $\phi_1 \vee \phi_2$.
- Spatial operators. e.g., $\exists_B \bigcirc \phi$, $\forall_B \bigcirc \phi$, where B is a nonempty subset of the set of directions $\{NW, NE, SW, SE\}$.

The spatial operators $\exists_B \bigcirc$ and $\forall_B \bigcirc$ are read as *there exists in directions B next* and *for all directions B next*, respectively. $\exists_B \bigcirc \phi$ is interpreted as follows. For at least one of the nodes located in the next level of the quad-tree labeled with one of the directions in B, ϕ must be satisfied. $\forall_B \bigcirc \phi$ specifies that for all such nodes ϕ must be satisfied. We demonstrate how TSSL can be used to express spatial patterns through an example.

Example 3: Consider a 4×4 checkerboard as illustrated in Fig. 3(a). White cells have a value of 1 and black cells have a value of zero. This pattern can be expressed as the following

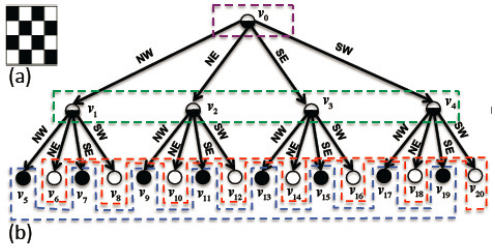

Fig. 3. Quad-tree representation of a 4 by 4 checkerboard pattern.

TSSL formula φ. The quad-tree of Fig. 3(b) satisfies φ.

$$\varphi = \forall_{\{NW,NE,SW,SE\}} \bigcirc [\forall_{\{NE,SW\}} \bigcirc (\mu_1 \geq 1) \wedge \forall_{\{NW,SE\}} \bigcirc (\mu_1 \leq 0)], \quad (11)$$

where μ_1 represents the mean value in a region expressed by spatial operators.

TSSL formulas can be viewed as pattern classifiers. For instance, the formula of equation 11 accepts a quad-tree derived from a checkerboard pattern and rejects any other quad-tree. Although TSSL is capable of describing complicated spatial behaviors in an image, it is difficult in general to write a formula that describes a complex pattern. In [12], the authors propose to use machine learning techniques in order to find such a formula from a given set of positive and negative examples.

Assume a set of positive images ($\mathcal{Y}_+$), illustrating a desirable pattern, and a set of negative images ($\mathcal{Y}_-$) are available. We can create set $\mathcal{L}$ from these images as

$$\mathcal{L} = \{(Q_y, +) \mid y \in \mathcal{Y}_+\} \cup \{(Q_y, -) \mid y \in \mathcal{Y}_-\}, \quad (12)$$

where Q_y is the quad-tree generated from image y. The set $\mathcal{L}$ is separated into a learning set $\mathcal{L}_L$ (used to train a classifier) and a testing set $\mathcal{L}_T$ (used to test the classifier obtained from $\mathcal{L}_L$) such that $\mathcal{L} = \mathcal{L}_L \cup \mathcal{L}_T$. A rules-based learner called RIPPER [20] is used to learn a set of classification rules from $\mathcal{L}_L$. The set of all rules learned by RIPPER are then translated into a single TSSL formula describing the pattern.

Example 4: Fig. 4 demonstrates sample images from a learning set created by the reaction-diffusion network of Example 1 (3). Fig. 4(a) shows a few images that resemble a maze-like pattern and belong to $\mathcal{Y}_+$. Fig. 4(b) shows a few other images belong to $\mathcal{Y}_-$. We created a total of 186 positive and 817 negative examples by simulating (3). 75% of the images were used for learning a TSSL descriptor for the maze pattern Φ_{maze} and the rest for testing. The learning procedure took 213 seconds on an iMac with 2.8 GHz Core i5 processor and 32 GB RAM and Φ_{maze} has 89% classification rate on the testing set. We cannot present Φ_{maze} here since it is very long. Instead, we show the part of Φ_{maze} corresponding to one of the RIPPER rules $\Phi_{\text{maze}}^{(1)}$ (There are 8 rules in all).

$$\Phi_{\text{maze}}^{(1)} = \exists_{NW}\exists_{SE}\exists_{NW}\exists_{NE}\exists_{NW} \bigcirc (\mu_2 \leq 0.002) \wedge \exists_{NW}\exists_{NE}\exists_{SW}\exists_{NW}\exists_{NW} \bigcirc (\mu_1 \geq 0.609) \wedge \exists_{NW}\exists_{NW}\exists_{NW} \bigcirc (\mu_2 \geq 0.008)$$

A formal recursive definition for the qualitative semantics of TSSL is presented in [12]. These semantics can be used to assign a true (satisfied) or false (violated) label to a TSSL specification with respect to an unlabeled quad-tree. While this is useful to check if an image demonstrates a desirable pattern or not, it does not provide any information about how strongly it satisfies or violates the given property. To provide information about how strongly an image satisfies or violates the given property, TSSL is also equipped with a recursive quantitative semantics definition which assigns a real value to a TSSL formula ϕ with respect to vertex $v \in \mathcal{V}$ of quad-tree $Q = (\mathcal{V}, R)$; denoted by $\rho(\phi, v)$. It is proven in [12] that TSSL quantitative semantics are sound. In other words, a quad-tree Q satisfies a formula ϕ ($Q \models \phi$) if $\rho(\phi, v_0) > 0$, where v_0 is the root of Q, and Q violates ϕ ($Q \not\models \phi$) if $\rho(\phi, v_0) < 0$. Therefore, the problem of checking whether an image contains a pattern expressed as a TSSL formula reduces to computing its quantitative value. Moreover, the absolute value of $\rho(\phi, v_0)$ can be viewed as a measure of how strongly ϕ is satisfied (or violated) by Q. Hence, the quantitative value of a formula with respect to a quad-tree is called its *robustness*.

At this point, we are able to quantify how strongly a quad-tree representation of an image satisfies or violates emergence of a pattern specified by a TSSL formula. This metric can serve as the fitness function in an optimization process over the parameter space Ω from (1). The goal is to determine the parameterization p_i^* that maximizes this metric on average:

$$p_i^* = \arg\max_{p \in \Omega} E(\rho(\Phi_i, v_0)), \tag{13}$$

where Φ_i is a TSSL formula that describes the pattern of cluster Y_i learned using the procedure that was explained earlier in this section. ρ is the robustness degree of a formula, and $E(\rho)$ is its expected value. We estimate the expectation by computing a sample mean

$$E(\rho(\Phi_i, v_0)) = \sum_{j=1}^{N_s} \rho(\Phi_i, v_0^{(j)}), \tag{14}$$

where N_s is the sample size and $v_0^{(j)}$ is the root of the jth sample quad-tree. The authors in [21] show that the expected robustness can usually be estimated by a relatively small sample size in practice.

Off the shelf optimization techniques can be used to solve (13). Inspired by [12] and [21], we employed particle swarm optimization (PSO) [22] to solve this problem. PSO is a heuristic solution to unconstrained optimization problems that is capable of solving problems with irregular search spaces and does not require the fitness function to be differentiable. As opposed to [12], [21] where PSO was initialized randomly, the clusters from Sec. III-B give us preliminary knowledge of parameters that can lead to desired patterns (10). These parameters can be used to initialize PSO particles, resulting in faster convergence.

At this point, the user is presented with the optimal solution p_i^* as well as samples of steady state behaviors that the optimal solution produces. If the resulting steady state images are satisfactory (i.e., resemble the desired pattern cluster Y_i well and frequently), the algorithm is terminated. Otherwise, we can conclude that the TSSL descriptor Φ_i does not represent the desired pattern well enough, which means that the learning set $\mathcal{L}_L$ used to learn Φ_i was not sufficient. In that case, we add the new images obtained from p_i^* to the set of negative images $\mathcal{Y}_-$, relearn Φ_i, and solve (13) with the new TSSL formula. This iterative procedure is continued until the steady state behaviors satisfy the expert.

IV. CASE STUDY

We consider the 32×32 reaction diffusion system described in Example 1 as a case study. The initial concentration of the species and the parameter search space are

$$\begin{aligned} &0 \leq x_{m,n}^{(i)}(0) \leq 16 \quad i = 1, 2, \\ &p = \langle D_1, D_2, R_1, R_2, R_3, R_4 \rangle \in \Omega, \\ &\Omega = [0, 10] \times [0, 30] \times \{1\} \times \{-12\} \times \{-1\} \times \{16\}. \end{aligned} \tag{15}$$

The authors in [12] used the formal methods approach explained in Sec. III-C to synthesize diffusion rates that lead to emergence of large spots, small spots, and fine patches illustrated in Fig. 1. However, they assume a learning set for each desired pattern is already given. This might not be a reasonable assumption since creating a learning set requires some knowledge of parameter values (diffusion and reaction rates) unless those images can be acquired through experiments on living cells. Moreover, system (3) might be capable of producing additional patterns that we do not have knowledge of beforehand. Here, we use the framework developed in Sec. III to study pattern emergence in the system without an already available labeled training set.

We start by exploring the state space and collecting steady state images. At every iteration, parameter values and initial conditions are randomly chosen from (15), and (3) is simulated five times using these values. If any of the five simulations reach steady state within 60 seconds, the corresponding steady state images are added to $\mathcal{Y}$. This process is continued until $|\mathcal{Y}| \geq 1000$.

Next, we performed the clustering procedure of Sec. III-B on $\mathcal{Y}$. The process took approximately 48 hours on a computing cluster with 16 processors at 2.1 GHz. Fig. 4(a) and 5(a) demonstrate sample images from two different nodes of the dendrogram. We call the patterns in Fig. 4(a) "maze" and the patterns in Fig. 5(a) "patches". Now, we intend to determine parameterizations p^*_{maze} and p^*_{patches} that result in frequent emergence of patterns as close to maze and patches as possible, respectively.

The details of learning a TSSL descriptor for the maze pattern (Φ_{maze}) was described in Example 4. We learned the patches TSSL formula Φ_{patches} similarly.

It took 46 minutes for the PSO algorithm to solve (13) for Φ_{maze}. We used 10 samples ($N_s = 10$) to compute (14) at each iteration. The optimal parameter values are

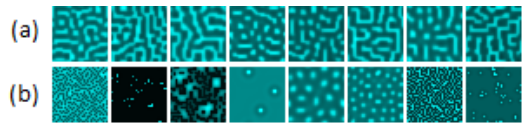

Fig. 4. (a) Samples from the set of 186 positive images from steady state observations belonging to the maze cluster (b) Samples from the set of 817 steady state observations not belonging to the maze cluster.

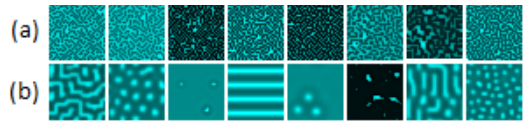

Fig. 5. (a) Samples from the set of 344 positive images from steady state observations belonging to the patch cluster (b) Samples from the set of 659 steady state observations not belonging to the patch cluster.

$$p^*_{\text{maze}} = \langle 2.9, 30, 1, -12, -1, 16 \rangle. \tag{16}$$

Fig. 6 illustrates three sample trajectories of (3) with the parameterization p^*_{maze} and different random initial conditions. It can be seen that the maze pattern emerges in all three instances.

The same procedure was performed for the patches pattern. It took 63 minutes for the PSO algorithm to solve (13) for Φ_{patches}. The optimal parameter values are

$$p^*_{\text{patches1}} = \langle 0.01, 2.9, 1, -12, -1, 16 \rangle. \tag{17}$$

Although the steady state patterns in this case are relatively close to the pattern we were looking for, they are a little darker and denser than what was expected. In order to investigate whether we can improve the results and get finer patches, we added 30 random examples of steady state images resulting from p^*_{patches1} to the set of negative examples $\mathcal{Y}_-$, relearned the TSSL formula for the new set, and recalculated the optimal parameters

$$p^*_{\text{patches2}} = \langle 0.3, 30, 1, -12, -1, 16 \rangle. \tag{18}$$

Fig. 7 illustrates three sample trajectories of (3) with the parameterization p^*_{patches2} and different random initial conditions. In all three cases, fine patches emerge.

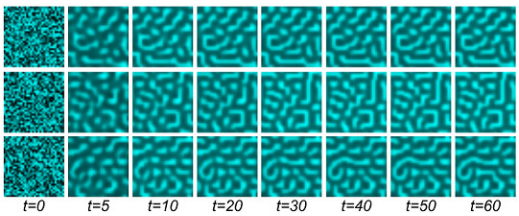

Fig. 6. Three sample trajectories of the Turing system (3) with p^*_{maze}.

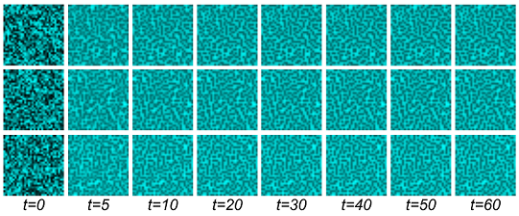

Fig. 7. Three sample trajectories of the Turing system (3) with p^*_{patches2}.

REFERENCES

[1] L. A. Montestruque and P. J. Antsaklis, "On the model-based control of networked systems," *Automatica*, vol. 39, no. 10, pp. 1837–1843, 2003.

[2] D. G. Green, "Emergent behaviour in biological systems," *Complex systems: from biology to computation*, pp. 24–35, 1993.

[3] M. B. Egerstedt and X. Hu, "Formation constrained multi-agent control," 2001.

[4] P. Bak, K. Chen, and M. Creutz, "Self-organized criticality in the game of life," *Nature*, vol. 342, no. 6251, pp. 780–782, 1989.

[5] M. A. Kinney, T. A. Hookway, Y. Wang, and T. C. McDevitt, "Engineering three-dimensional stem cell morphogenesis for the development of tissue models and scalable regenerative therapeutics," *Annals of biomedical engineering*, vol. 42, no. 2, pp. 352–367, 2014.

[6] A. Shkumatov, K. Baek, and H. Kong, "Matrix rigidity-modulated cardiovascular organoid formation from embryoid bodies," *PloS one*, vol. 9, no. 4, p. e94764, 2014.

[7] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Multi-robot system for artistic pattern formation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4512–4517.

[8] A. S. Rufino Ferreira and M. Arcak, "A graph partitioning approach to predicting patterns in lateral inhibition systems," *SIAM Journal on Applied Dynamical Systems*, vol. 12, no. 4, pp. 2012–2031, 2013.

[9] C. M. Bishop, "Pattern recognition," *Machine Learning*, vol. 128, 2006.

[10] D. Briers, I. Haghighi, D. White, M. L. Kemp, and C. Belta, "Pattern synthesis in a 3d agent-based model of stem cell differentiation," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 4202–4207.

[11] R. Grosu, S. A. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, and E. Bartocci, "Learning and detecting emergent behavior in networks of cardiac myocytes," *Communications of the ACM*, vol. 52, no. 3, pp. 97–105, 2009.

[12] E. Bartocci, E. A. Gol, I. Haghighi, and C. Belta, "A formal methods approach to pattern recognition and synthesis in reaction diffusion networks," *IEEE Transactions on Control of Network Systems*, 2016.

[13] A. M. Turing, "The chemical basis of morphogenesis," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 237, no. 641, pp. 37–72, 1952.

[14] T. Guha and R. K. Ward, "Image similarity using sparse representation and compression distance," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 980–987, 2014.

[15] Z. Zhang, Y. Chen, and V. Saligrama, "A novel visual word co-occurrence model for person re-identification," in *European Conference on Computer Vision*. Springer, 2014, pp. 122–133.

[16] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," *Journal of Machine Learning Research*, vol. 11, no. Mar, pp. 1109–1135, 2010.

[17] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications*. Springer Science & Business Media, 2009.

[18] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *PROCEEDINGS-IEEE*, vol. 95, no. 1, p. 138, 2007.

[19] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[20] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the twelfth international conference on machine learning*, 1995, pp. 115–123.

[21] I. Haghighi, A. Jones, Z. Kong, E. Bartocci, R. Gros, and C. Belta, "Spatel: a novel spatial-temporal logic and its applications to networked systems," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 189–198.

[22] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.