2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Spatial-Temporal Pattern Synthesis in a Network of Locally Interacting Cells

Noushin Mehdipour[1], Demarcus Briers[1], Iman Haghighi[1], Chad M. Glen[2], Melissa L. Kemp[2] and Calin Belta[1]

Abstract— **Spatial-temporal patterning is an essential process indicating the cell and tissue types formed during synthetic and natural tissue development. Due to biological complexity within and among cells, most approaches to engineer cell cultures are limited in their ability to control the timing of spatial patterning. This paper presents a computational framework to control spatial-temporal pattern formation in a network of locally interacting agents. Given a parameterized model of patterning in stem cell colonies, we use a supervised machine learning algorithm combined with temporal logics to quantitatively describe and characterize spatial-temporal patterns. We utilize an optimization procedure to achieve optimal parameters that maximize the occurrence of desired spatial patterns at a specified time interval and demonstrate the ability of our algorithm to control transitioning in a sequence of patterns.**

I. INTRODUCTION

Pluripotent Stem Cells (PSCs) have the unique ability to convert into different cell types and are commonly employed to study early human development. Recently, PSCs have been transformed into lab-grown organoids that can resemble the healthy or diseased organ systems [1]. As a result, PSCs have become an essential biological tool in the quest to engineer organoids formation and regenerate human tissues. The conversion of PSCs into specialized cell types by losing pluripotency, known as differentiation, typically begins with a homogeneous stem cell population. Every PSC then receives a chemical cue that induces spontaneous differentiation at different locations within a colony [2]. This process often occurs asynchronously in the population, producing cells that differentiate in spatial patterns that evolve over several days [3], [4]. The sequence of transient spatial patterns that emerge due to differentiation often dictates the type and diversity of the resulting tissues. Therefore, characterizing and controlling the timing and sequence of differentiation patterns in stem cells is a crucial step toward more precise engineered living systems.

Due to the complex interactions and stochasticity in the differentiation dynamics, it is difficult to predict and control this process without the aid of computational approaches. There have been several efforts to build mathematical models describing differentiation dynamics in stem cells [5], [6]. It was shown in [6] that local interactions between stem cells might explain spatial pattern formation during differentiation. In [3], [4], authors developed an agent-based model where the diffusion of chemical signals regulates the emergence of spatial patterns in a three-dimensional spheroid of stem cells. Authors in [7] improved the previous agent-based approach by modeling changes in the permeability of cells and discovered a novel mechanism to perturb spatially resolved differentiation. These prior studies have discovered novel mechanisms that accurately model and predict the evolution of spatial differentiation patterns in stem cells. However, attempts to induce steady-state patterns in stem cell populations relied on brute-force approaches to test every parameter combination [4]. Searching through the entire parameter space is computationally expensive for a large number of parameters, especially since these models are computationally costly. Therefore, simulation-based optimization methods can be used to enhance the search for optimal tunable parameters [8]. In this paper, we propose a formal methods framework capable of quantifying how strongly spatial patterns and trajectories of spatial patterning match user-specifications. We use an optimization algorithm to efficiently explore the parameter space and determine optimal tunable parameters that maximize the closeness of spatial-temporal patterning to the given specification.

Formal methods have been widely studied to solve control problems under complex high-level specifications [9]. Formal logics with time constraints such as Signal Temporal Logic (STL) [10] allow us to define tasks with temporal deadlines in a formal specification language. STL is a purely temporal logic capable of specifying signal properties (e.g., "The signal value must always be positive and eventually become over 5 within 10 seconds") and can be used to monitor system behavior. Therefore, there have been significant efforts to infer temporal logic formulas from data to discriminate between desirable and undesirable system behaviors, either by estimating parameters in a given structure or solving the structure inference and parameter estimation simultaneously [11], [12]. Recently, there have been efforts to use formal logics to specify spatial properties in spatially distributed networks. Linear Spatial Superposition Logic (LSSL) [13] and Tree Spatial Superposition Logic (TSSL) [14] are a few examples of such efforts that could only capture spatial properties. A temporal extension of TSSL, called SpaTeL, was introduced in [15]. This logic was a unification of TSSL and STL and capable of describing how spatial properties evolve. Signal Spatio-Temporal Logic (SSTL) [16] and Spatio-Temporal Reach and Escape Logic (STREL) [17] are other spatial extensions to signal temporal logic. These logics

*This work was partially supported at Boston University by NSF under grants CNS-1446607,CBET-0939511 and at Georgia Institute of Technology under grant CBET-0939511 and a graduate research fellowship to C.M.G. from the Natural Sciences and Engineering Research Council of Canada.

Noushin Mehdipour, Demarcus Briers, Iman Haghighi and Calin Belta {noushinm,dbriers,haghighi,cbelta}@bu.edu are with the Boston University, Boston, MA, USA. Chad Glen cglen@gatech.edu and Melissa Kemp melissa.kemp@bme.gatech.edu are with the Georgia Institute of Technology and Emory University, Atlanta, GA, USA.

are much more complicated than pure temporal logics such as STL since they are designed to capture properties in both space and time, making it more difficult to verify whether specifications expressed in these logics are satisfied or not.

In this paper, we demonstrate that STL can be utilized to describe a wide variety of spatial-temporal properties without using more complicated spatial logics. Inspired by [14], we employ a machine learning algorithm to generate STL descriptors with spatial information inferred from data. The STL descriptor is only satisfied when system execution conforms to the specified sequence of the patterns present in the training data. Additionally, the STL quantitative semantics is used to reduce the dynamic pattern synthesis problem into a single optimization problem. Our proposed framework can be used to monitor and control spatial-temporal patterning in any system with time-evolving spatial patterns. We will demonstrate the applicability of this framework in the problem of synthesizing spatial-temporal patterns in an agent-based model of stem cell differentiation and show that patterns are synthesized in a more computationally efficient way by employing simple STL descriptors to define dynamic patterns rather than complex spatial-temporal logics.

II. PRELIMINARIES

A. Signal Temporal Logic (STL)

STL has been used to monitor temporal properties of real-value signals. Consider a discrete time sequence $\tau := \{t'_k | k \in \mathbb{Z}_{\geq 0}\}$. A *trace* or *signal* σ is a function $\sigma : \tau \to \mathbb{R}^N$ that maps each time point $t \in \tau$ to a N-dimensional vector of real values $\sigma(t)$, with $x_i(t)$ being its ith component, i.e. $\sigma(t) = [x_1(t), ..., x_N(t)]'$. A specification written in STL consists of *predicates* of the form $\mu := l(\sigma) \geq 0$, where l is a real function defined over values of one or more elements of σ (e.g., $x_1 - 2 \geq 0$ or $x_1^2 - x_3^3 + 1 \geq 0$); as well as Boolean operators $(\neg, \wedge, \vee)$ and temporal operators $(\mathbf{F}, \mathbf{G})$ [10]. We denote $[t_1, t_2] = \{t' | t', t_1, t_2 \in \tau; t_1 \leq t' \leq t_2;\ \ t_2 > t_1 \geq 0\}$. The temporal operator *Finally* ($\mathbf{F}_{[t_1,t_2]}\varphi$), also called *eventually*, means that "at some time point in the assigned interval the specification φ must be satisfied"; while *globally* ($\mathbf{G}_{[t_1,t_2]}\varphi$), also called *always*, means that "φ must be satisfied at all times within the interval".

STL is equipped with qualitative semantics showing *whether* a specification φ with respect to trace σ at time t is satisfied or not, and quantitative semantics $\rho(\varphi, \sigma, t)$, also known as *robustness*, that measures *how much* it is satisfied or violated. Given a predicate μ, a trace σ and specifications φ and ψ, the robustness is recursively computed as [10]:

$$\begin{aligned}
&\rho(\mu, \sigma, t) := l(\sigma(t)), \\
&\rho(\neg\varphi, \sigma, t) := -\rho(\varphi, \sigma, t), \\
&\rho(\varphi \wedge \psi, \sigma, t) := \min(\rho(\varphi, \sigma, t), \rho(\psi, \sigma, t)), \\
&\rho(\varphi \vee \psi, \sigma, t) := \max(\rho(\varphi, \sigma, t), \rho(\psi, \sigma, t)), \\
&\rho(G_{[t_1,t_2]}\varphi, \sigma, t) := \min_{k \in [t+t_1, t+t_2]} \rho(\varphi, \sigma, k), \\
&\rho(F_{[t_1,t_2]}\varphi, \sigma, t) := \max_{k \in [t+t_1, t+t_2]} \rho(\varphi, \sigma, k),
\end{aligned} \tag{1}$$

which is a real number quantifying the degree of satisfaction of the specification. Robustness degree is sound, meaning that a trace σ satisfies φ at time t if $\rho(\varphi, \sigma, t) > 0$ and violates it if $\rho(\varphi, \sigma, t) < 0$. A higher robustness degree corresponds to a stronger satisfaction of the specification. Consequently, larger disturbance in the signal would violate a specification with a higher robustness score. Therefore, it is desirable to maximize robustness for a given specification. With a slight abuse of notation, we denote the robustness degree of φ at time 0 with respect to the trace σ by $\rho(\varphi, \sigma)$.

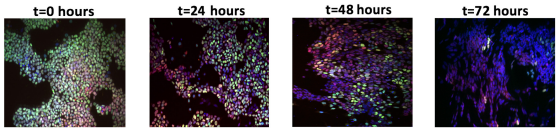

Fig. 1. Microscopy images of stem cells at different time steps. The initial population of all pluripotent stem cells uniformly expressing Oct4 (cyan) in left. Transitioning patterns observed during differentiation expressing Sox2 (red) in the middle. All differentiated stem cells after 3 days, nuclei stained with Hoechst (blue) and showing loss of cyan Oct4 expression in the right.

B. Binary Classification

Among different machine learning algorithms for classification, Support Vector Machine (SVM) is widely used with biological systems due to its high speed and accuracy in multi-dimensional space [18]. Given a set of labeled training data points in $\mathbb{R}^N$, SVM finds the best hyperplane to separate different classes (data points with different labels). In the case where the distribution of data is linearly separable, linear SVM is preferable due to its high accuracy and fast training.

Consider a 2 class classification problem with a set of n training samples. Each sample is indicated by a tuple $[(X_i, Y_i)]$, where $X_i \in \mathbb{R}^N$ is the i^{th} sample and $Y_i \in \{-1, 1\}$ is its associated label with $Y_i = 1$ representing positive samples and $Y_i = -1$ representing negative ones. Linear SVM finds the decision boundary $\omega^T X + b = 0$ by maximizing the geometric margin between support vectors and the resulting classifier is:

$$f(X) = \text{sign}\left(\omega^T X + b\right). \tag{2}$$

Using (2), one can predict to which class a new data point belongs, i.e. $f(X)$ classifies a test sample $X \in \mathbb{R}^N$ as a member of the positive class if $\omega^T X + b > 0$ and the negative class if $\omega^T X + b < 0$. Moreover, the Euclidean distance γ_i of a sample X_i from the decision boundary is defined as:

$$\gamma_i = \frac{Y_i\left(\omega^T X_i + b\right)}{\|\omega\|}. \tag{3}$$

Samples with larger distance from the decision hyperplane are classified more accurately, while there is uncertainty in classifying samples closer to the boundary.

III. STOCHASTIC AGENT-BASED MODEL

Building a computational model of differentiation allows us to perform quantitative studies and easily manipulate patterns by modifying model parameters. In [7], a model of a novel differentiation mechanism was developed in which the production and intercellular diffusion of metabolites,

represented by the molecule cyclic Adenosine Monophosphate (cAMP), drives the distribution of spatial patterns in an experimental system of initial symmetry breaking. This molecule has been associated as a driver of differentiation and serves as an illustrative example of intercellular molecular control over differentiation patterning. A set of microscopy images of stem cells differentiation observed in actual experiments is shown in Fig. 1. Stem cells can be in one of the two differentiation states: Undifferentiated (Pluripotent) or Differentiated. All cells are initially undifferentiated and divide asynchronously by an irreversible change in cell state (differentiate) due to the local dynamic interactions and the accumulation of metabolites.

Modeling and controlling the stochastic spatial-temporal dynamics of stem cell differentiation poses a significant challenge for many experimentalists. In our 2 dimensional agent-based model, a network of $N(t)$ locally interacting stem cells at time t is considered. Cells are labeled by an integer $i \in \{1, \cdots, N(t)\}$ and network is represented by a graph with vertex set containing all cells at time t and edge set indicating which cells are adjacent neighbors. The concentration of the metabolite c_i in the stem cell i is determined by the production and degradation of the diffusible molecule in that cell, the differentiation state of the cell and the progression of the cell in its cell division cycle. Changes in the concentration of the metabolite in the cell i is described as:

$$dc_i/dt = P(c_i) - D(c_i) + E(c_i),$$

where $P(c_i)$ is the production, $D(c_i)$ is the degradation and $E(c_i)$ is the exchange of metabolites due to the gradient-driven diffusion between adjacent cells. The production of the metabolite in the cell i is defined by the Hill Function:

$$P(c_i) = \frac{\alpha}{1 + (c_i/\beta)^n}, \tag{4}$$

where α is the maximum production rate, β represents the concentration of metabolite in which the production rate saturates to half of its maximal rate, and n controls the steepness of the production rate saturation. The degradation of the metabolite in the cell i is described by the proportional relationship:

$$D(c_i) = d\, c_i, \tag{5}$$

where d is the degradation rate. The change in the metabolite concentration due to the diffusion between cell i and its adjacent neighbors is described as:

$$E(c_i) = \sum_{j \in Neighbors_i} F(i,j,t)(c_j - c_i),$$

where $F(i,j,t)$ is an asynchronous sigmoid function interpreted as permeability efficiency between neighboring cells i and j at time t and $c_j - c_i$ represents how molecules passively diffuse from high to low concentration between cells.

The described model is implemented in Python with stem cells modeled as incompressible circles initially in a dense, irregularly shaped colony with a population of $100 - 1000$ cells, which is the geometry and colony size seen in many biological studies of stem cell differentiation [3], [4]. All

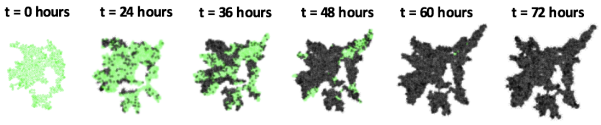

Fig. 2. A sample trace of differentiation in simulation. Green and black circles represent undifferentiated and differentiated cells, respectively.

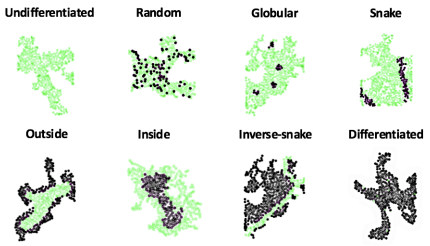

Fig. 3. Different patterns formed due to differentiation in stem cells.

stem cells are initially undifferentiated (colored green) and transition into a differentiated state (colored black) as illustrated in Fig. 2. Each time step in simulation takes about 2 minutes and mimics the differentiation occurring in experiments in 1 hour. α, β and d in (4), (5) for both differentiated (D) and undifferentiated (U) cells are the control parameters and other parameters are forced by the system.

IV. FEATURE EXTRACTION AND PATTERN CLASSIFICATION

Different patterns emerge based on when and where differentiation occurs in the cell population. In [7], these different patterns are classified into 8 classes, shown in Fig. 3. Differentiation patterns may resemble random spots, globular shaped clusters, snake-like clusters, differentiation in outside or inside of the colony, or an inverse-snake shape. To describe and characterize these different patterns, we extract quantitative features based on their spatial properties. A supervised machine learning algorithm is then used to classify patterns in their multi-dimensional feature space. These trained classifiers can be later used to measure the level of satisfaction of a desired spatial specification.

A. Patterns and Features

To describe the variations in the spatial patterns illustrated in Fig. 3, we extract 9 network and subnetwork spatial features including total percentage of differentiation, peripheral and central percentage of differentiation, average and standard deviation of the circularity of the differentiated clusters, average radial distance of undifferentiated and differentiated clusters, path length of the largest undifferentiated cluster with respect to the path length of the whole colony and number of differentiated subclusters with less than 4 cells. We generate a dataset of 8000 samples (1000 samples for each pattern) and extract these 9 features to map patterns into a feature space. To visualize the 9-dimensional dataset, we use Linear Discriminant Analysis (LDA) to map data to a lower dimension. In Fig. 4, LDA with 3 components is

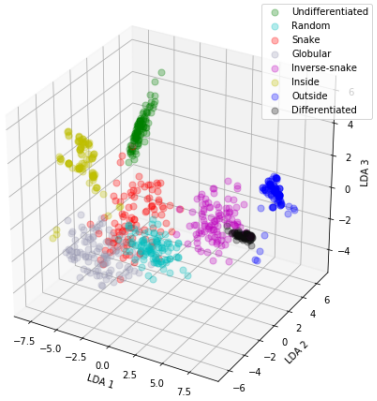

Fig. 4. Visualization of patterns reduced from 9 to 3 dimension using LDA

applied to our dataset, which covers 89% of the variance of data with first, second and third component (LDA1, LDA2, LDA3) covering 57%, 20% and 12% of data variation, respectively. This reduced 3D visualization helps us to realize that some of the classes are partially linearly separable.

B. Multi-class Classification

To classify all the different patterns in Fig.3, we use the One-VS-All classification approach [19]. One-Vs-All Linear SVM is a supervised multi-class classification algorithm which uses the same idea as the binary classification described in Sec. II and fits one classifier for each class against all the other classes. We first use a grid search technique to tune parameter C, known as SVM penalty parameter of the error, which trades off between misclassification and how large the margin between classes is. After tuning C, a linear SVM classifier is learned for each pattern in the multi-dimensional feature space. Classification accuracy for our dataset of 8000 samples with 9 attributes (divided to 6400 train samples and 1600 test samples) was 97.5% on the test set, with both precision and recall equal to 98%.

V. FORMAL SPECIFICATION OF SPATIAL-TEMPORAL BEHAVIORS

To synthesize spatial time-varying patterns, we need to control the spatial properties, as well as the timing and sequencing of emergence of patterns. We can use the trained classifiers in Sec. IV to check if a simulation execution resembles the desired pattern or not using (2), and measure how far it is from the boundary hyperplane that separates patterns using (3). This geometric distance can be used as the spatial quantitative valuation to score the level of satisfaction of desired spatial patterns. Moreover, STL can be used as a tool to specify the system temporal requirements. Thus, in order to describe spatial properties of the system over time, we nest spatial characteristics in STL formulas by defining predicates μ as:

$$\mu_i := \frac{{\omega_i}^T X + b_i}{\|\omega_i\|}, \tag{6}$$

where $i \in \{$*Undifferentiated, Random, Globular, Snake, Outside, Inside, Inverse-snake, Differentiated*$\}$ and X is the 9 dimensional vector of extracted features. The desired spatial pattern i is obtained if ${\omega_i}^T X + b_i > 0$ and is not if ${\omega_i}^T X + b_i < 0$. This definition of μ conforms with the soundness of STL, meaning that for a trace σ of patterns generated by the system, $\rho(\mu_i, \sigma) > 0$ if the desired spatial-temporal specification is satisfied and $\rho(\mu_i, \sigma) < 0$ if it is violated. Therefore, by embedding spatial predicates in STL, we can specify spatial behaviors of the system over time. For instance, we can specify the spatial-temporal specification "at some time within 30 hours reach the *Snake* pattern" as a STL formula: $\varphi = F_{[0,30]}\mu_{Snake}$. The spatial-temporal quantitative valuation of a dynamic trajectory σ (which is the 9-dimensional extracted feature vector) generated by system is computed combining (1), (6):

$$\rho(\varphi, \sigma) = \max_{k \in [0,30]} \left(\frac{\omega_{Snake}^T X(k) + b_{Snake}}{\|\omega_{Snake}\|}\right).$$

Using this spatial-temporal robustness score, we can monitor whether a trace generated in simulation satisfies the desired spatial-temporal specification, and quantify its level of satisfaction. As discussed earlier, higher robustness degree corresponds to a stronger satisfaction of the specification.

VI. OPTIMIZATION

Using the computational model, we can easily apply changes in simulations and see how the resulting patterns change. Tunable parameters in our system are:

$$\Pi = (\alpha_U, \beta_U, d_U, \alpha_D, \beta_D, d_D),$$

which vary in the range space Ω:

$$\Omega = \Omega_{\alpha_U} \times \Omega_{\beta_U} \times \Omega_{d_U} \times \Omega_{\alpha_D} \times \Omega_{\beta_D} \times \Omega_{d_D},$$

where Ω_i is the range space for parameter i. Each simulation takes approximately 2 hours on average on a machine with a 2.5GHz Core i7 CPU and 16 GB RAMs to run. This is much faster than doing experiments in lab, which normally takes 72 hours for one experiment. However, it is still time-consuming and not efficient to randomly explore the parameter space to find pattern producing parameters. Thus, in order to efficiently explore the parameter space Ω to find the best environmental conditions leading to the emergence of desired patterns, we propose to maximize the spatial-temporal quantitative robustness degree described in Sec. V. This robustness degree can be used as a reward function in an optimization problem:

$$\Pi^* = \arg\max_{\Pi \in \Omega} \rho(\varphi_{\text{pattern}}, \sigma). \tag{7}$$

The optimal tunable parameters Π^* provide promising environmental conditions that ensure emergence of the desired patterns at the desired time. To solve (7) which has a non-differentiable fitness function, we use a heuristic optimization algorithm called Particle Swarm Optimization (PSO), which iteratively evaluates the fitness function for different parameters until it finds the optimal solution [20]. PSO begins by initializing each particle of the swarm with a random position $p_i \in \Omega$ and velocity v_i which defines the direction from the current position toward the optimal solution. At each iteration of PSO, we run a simulation for each particle in the swarm

to generate the resulting trace. PSO then evaluates the fitness function to determine the current best particle position p_i^{best} and current best swarm position p^s, and updates velocities to move particles to their new positions toward the optima:

$$v_i \leftarrow \eta v_i + \phi_p r_p \left(p_i^{best} - p_i\right) + \phi_s r_s \left(p^s - p_i\right),$$
$$p_i \leftarrow p_i + v_i,$$

where r_p, r_s are random numbers $r_p, r_s \sim (0,1)$ and η, ϕ_p and ϕ_s are user-defined parameters. The termination criterion is met either if p^s does not change after some consecutive iterations or the maximum number of iterations is reached.

VII. CASE STUDY

We test the proposed framework for three spatial-temporal specifications defined in STL formulas as follows:

Case 1: $\varphi_1 = F_{[t_1,t_2]} G_{[0,T]} \mu_{Outside}$

which is interpreted as "*eventually* in $[t_1, t_2]$ hours reach pattern *Outside* and *always* remain in it for the next T hours". Robustness degree of this formula given a simulation execution σ is recursively computed using (1), (6):

$$\rho(\varphi_1, \sigma) = \max_{k' \in [t_1,t_2]} \left(\min_{k \in [k', k'+T]} \left(\frac{\omega_{Outside}^T X(k) + b_{Outside}}{\|\omega_{Outside}\|} \right) \right).$$

Case 2: $\varphi_2 = F_{[t_1,t_2]} G_{[0,T]} \mu_{Inside} \wedge G_{[0,t_2]} \neg \mu_{Random}$

which is interpreted as "*eventually* in $[t_1, t_2]$ hours reach pattern *Inside* and *always* stay in that pattern for the next T hours *and always* avoid *Random* pattern in $[0, t_2]$ hours".

Case 3: $\varphi_3 = F_{[t_1,t_2]} G_{[0,T]} \mu_{Globular} \wedge F_{[t_3,t_4]} G_{[0,T']} \mu_{Snake} \wedge F_{[t_5,t_6]} G_{[0,T'']} \mu_{Inverse\text{-}snake}$

which is a sequence of patterns interpreted as "*eventually* in $[t_1, t_2]$ hours reach pattern *Globular* and *always* remain in it for the next T hours *and eventually* in $[t_3, t_4]$ hours reach pattern *Snake* and *always* stay in it for the next T' hours *and eventually* in $[t_5, t_6]$ hours reach pattern *Inverse-snake* and *always* remain in it for the next T'' hours". Robustness degree for φ_2 and φ_3 can be derived in a similar way as φ_1.

PSO needs a large population of particles when there are many parameters to optimize. Therefore, we decided to tune 2 biologically interpretable parameters α_D and β_D for differentiated cells with $\Omega_{\alpha_D} = \left[e^{-6}, e^{-2}\right]$ and $\Omega_{\beta_D} = [0,1]$ and fixed the others. Due to non-determinism and stochasticity in our system, traces produced for same parameters may be different. We therefore determine parameters Π^* such that the resulting traces, on average, satisfy desired specifications. We maximize the expected value of robustness in (7) by calculating the sample mean of robustness of traces generated for each parameter set. The PSO optimization was distributed on a cluster with 16 processors at 2.1GHz. Having 16 particles randomly distributed in the search space, run time was about 19 hours for φ_1 and optimized parameters found $\alpha_D^* = 8.37\text{e}^{-6}, \beta_D^* = 0.046$; 13 hours for φ_2 with $\alpha_D^* = 4.99\text{e}^{-3}, \beta_D^* = 0.037$; and 15 hours for the sequence of patterns in φ_3 with $\alpha_D^* = 4.20\text{e}^{-6}, \beta_D^* = 0.835$. Traces generated with these optimal parameters in an initial colony with 365 cells are shown in Fig. 5. In Fig. 5(a), *Outside* pattern is reached at $t = 27$ hours and kept for next 2 hours. Fig. 5(b) shows that *Inside* pattern emerges at $t = 3$ hours and is held for next 3 hours while always avoiding *Random* pattern. In Fig. 5(c), a sequence of *Globular*, *Snake* and *Inverse-snake* patterns is observed, satisfying the spatial-temporal constraints in φ_3. Our results show that, strikingly, spatial-temporal patterning for three desired pattern sequences is achieved by modulation of only two parameters α_D and β_D in the metabolite production hill function. Therefore, the proposed approach advances the goal of engineering multicellular systems by identifying experimental perturbations that may yield desired emergent behaviors.

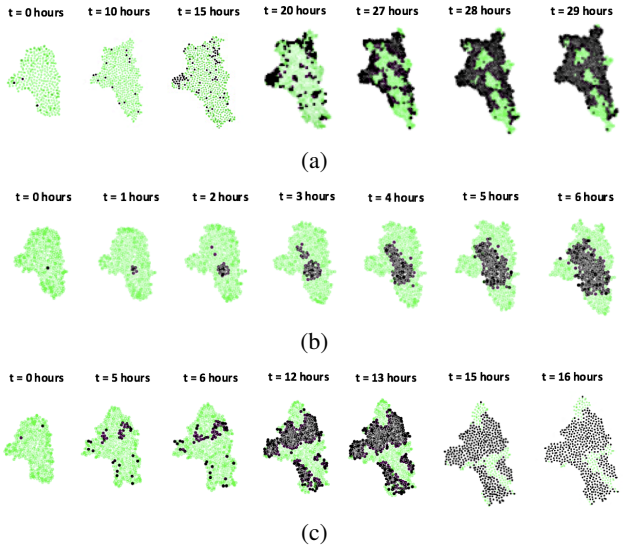

Fig. 5. Traces generated for optimal parameters satisfying specifications:(a) $\varphi_1 = F_{[20,40]} G_{[0,2]} \mu_{Outside}$,(b) $\varphi_2 = F_{[0,10]} G_{[0,3]} \mu_{Inside} \wedge G_{[0,10]} \neg \mu_{Random}$,(c) $\varphi_3 = F_{[5,10]} G_{[0,1]} \mu_{Globular} \wedge F_{[10,15]} G_{[0,1]} \mu_{Snake} \wedge F_{[15,20]} G_{[0,1]} \mu_{Inverse\text{-}snake}$.

This paper is closely related to [21], which presents a formal framework to synthesize steady-state spatial patterns in stem cell populations, without any constraints on temporal behaviors. Authors used TSSL to formally describe and quantify desirable spatial patterns. TSSL is defined over a complex quadtree data structure of the partitioned image and learns classifiers using the RIPPER learning algorithm. However, we propose to use STL to describe both spatial and temporal properties at the same time by defining predicates over spatial features and train linear SVM classifiers over the interpretable multi-dimensional feature space. Our work outperforms [21] in the sense that we solve dynamic pattern synthesis with temporal constraints (not just the steady-state) which allows us to monitor and control the emergence of sequences of patterns. We also compare these two methods to show the accuracy and efficiency of our approach.

Learning classifiers in TSSL is very time consuming since classifiers are trained separately for each pattern while in an One-VS-All SVM, classifiers for all patterns are trained simultaneously. Moreover, TSSL needs to find quadtrees which are recursively constructed by partitioning the image into quadrants. It is proved in [15] that increasing the depth of TSSL quadtree increases the learning time exponentially. We use TSSL over quadtrees with a depth of 5, and compare the classification accuracy and time of learning TSSL versus

TABLE I

CLASSIFICATION ACCURACY AND RUN TIME COMPLEXITY OF SVM AND TSSL FOR *Snake* PATTERN

Train Set		Test Set		Classification Rate		Learning Time	
Positive	Negative	Positive	Negative	SVM	TSSL	SVM	TSSL
100	700	20	140	97.5 %	93.2 %	0.025 sec.	1.5 sec.
1000	7000	200	1400	98.1 %	96.7 %	1.25 sec.	195 sec.
5000	35000	1000	7000	98.9%	98.3%	8.5 sec.	7695 sec.

linear SVM classifier for the *Snake* pattern in three different datasets with positive (*Snake* pattern) and negative (other patterns) samples. As illustrated in Table I, to train the TSSL classifier, we need a larger train set in order to get a high accuracy. Moreover, training time in TSSL increases significantly with the size of train set. However, linear SVM in a 9-dimensional feature space gives us a high accuracy even in small train sets, and learning the classifier is faster.

Computing spatial robustness score using TSSL is more complex than using linear SVM classifier described in this paper. To calculate TSSL quantitative valuation, one needs to save individual images, construct the quadtree representation from each image and calculate the spatial robustness degree using a usually long list of complex rules learned by RIPPER algorithm [14]. The complexity of constructing quadtrees grows exponentially with the image size. On the other hand, it is more intuitive to implement and score the robustness valuation using the SVM geometric margin in a biologically interpretable feature space. Moreover, these features are independent of the colony size and shape and can be generalized to other colonies, while TSSL uses images to train classifiers and needs large training sets for different colonies. Therefore, our method allows us to synthesize spatial patterns in a faster and more intuitive way. We are also able to add temporal constraints to specify and produce sequences of patterns rather than merely steady-state patterns.

VIII. CONCLUSIONS

We present a computationally efficient framework to synthesize spatial-temporal patterns using signal temporal logic. Our approach utilizes supervised machine learning techniques to classify patterns in the extracted multi-dimensional spatial feature space. We use the trained classifiers to measure how strongly a spatial property is achieved. Combined with temporal constraints, we specify desired spatial-temporal behaviors as STL formulas with which we can monitor dynamic spatial patterning over time. Using the proposed spatial-temporal quantitative robustness score as the fitness function, we run an optimization algorithm to find optimal parameters that maximize occurrence of desired spatial-temporal patterns. We apply the proposed framework to the spatial-temporal pattern synthesis problem in embryonic stem cell differentiation. Our approach allows us to monitor the emergence of desired spatial patterns at desired times, find environmental conditions to control what sequences of patterns to emerge, and hypothesize about biological mechanisms that can control the appearance of new patterns or sequences of patterns.

REFERENCES

[1] A. Fatehullah, S. H. Tan, and N. Barker, "Organoids as an in vitro model of human development and disease," *Nature Cell Biology*, vol. 18, no. 3, pp. 246–254, 2016. [Online]. Available: http://dx.doi.org/10.1038/ncb3312

[2] J. A. Montero and C. P. Heisenberg, "Gastrulation dynamics: Cells move into focus," *Trends in Cell Biology*, vol. 14, no. 11, pp. 620–627, 2004.

[3] D. E. White, M. A. Kinney, T. C. McDevitt, and M. L. Kemp, "Spatial pattern dynamics of 3d stem cell loss of pluripotency via rules-based computational modeling," *PLoS computational biology*, vol. 9, no. 3, p. e1002952, 2013.

[4] D. E. White, J. B. Sylvester, T. J. Levario, H. Lu, J. Todd Streelman, T. C. McDevitt, and M. L. Kemp, "Quantitative multivariate analysis of dynamic multicellular morphogenic trajectories," *Integrative Biology*, vol. 7, no. 7, pp. 825–833, 2015.

[5] P. Pir and N. Le Novère, *Mathematical Models of Pluripotent Stem Cells: At the Dawn of Predictive Regenerative Medicine*. New York, NY: Springer New York, 2016, pp. 331–350.

[6] Q. Smith, E. Stukalin, S. Kusuma, S. Gerecht, and S. X. Sun, "Stochasticity and spatial interaction govern stem cell differentiation dynamics," *Scientific reports*, vol. 5, p. 12617, 2015.

[7] C. M. Glen, T. C. McDevitt, and M. L. Kemp, "Dynamic intercellular transport modulates the spatial patterning of differentiation during early neural commitment," *Nature Communications, in press*, 2018.

[8] A. Gosavi *et al.*, "Simulation-based optimization," *parametric optimization techniques and reinforcement learning*, 2003.

[9] H. Kamp and U. Reyle, *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*. Springer Science & Business Media, 2013, vol. 42.

[10] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.

[11] Z. Kong, A. Jones, A. Medina Ayala, E. Aydin Gol, and C. Belta, "Temporal logic inference for classification and prediction from data," in *Proceedings of the 17th international conference on Hybrid systems: computation and control*. ACM, 2014, pp. 273–282.

[12] L. Nenzi, S. Silvetti, E. Bartocci, and L. Bortolussi, "A robust genetic algorithm for learning temporal specifications from data," in *International Conference on Quantitative Evaluation of Systems*. Springer, 2018, pp. 323–338.

[13] 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.

[14] 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.

[15] 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.

[16] L. Bortolussi and L. Nenzi, "Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic," in *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*. ICST, 2014, pp. 66–73.

[17] E. Bartocci, L. Bortolussi, M. Loreti, and L. Nenzi, "Monitoring mobile and spatially distributed cyber-physical systems," in *Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design*. ACM, 2017, pp. 146–155.

[18] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[19] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[20] M. E. H. Pedersen, "Good parameters for particle swarm optimization," *Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001*, 2010.

[21] 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.