# A Self-Organizing Map Based Approach to Adaptive System Formation

**Dizhou Lu and Yan Jin**
*University of Southern California, USA*

Multi-agent systems are considered to be potential solutions to complex tasks. Cellular self-organizing (CSO) multi-agent systems have been proposed that take a field-based approach to regulate agent behaviors. One difficulty in designing CSO systems is to generate rules to map given tasks to agent behaviors. This paper proposes an approach for adaptive system formation based on a field analysis and self-organizing map (SOM) algorithm. The tasks are captured as multiple task fields. The relationship among the agents is translated into a social field. Each agent has multiple function modes corresponding to the task fields. SOM and a function mode selection algorithm are devised to match the social field of the system with the task fields. Computer simulations have demonstrated the effectiveness of this approach and its potential in designing CSO systems for solving system formation tasks.

## Introduction

When changes in the task requirement and operation environment occur, a system often needs to change its formation (e.g., form/shape, size, or structure) in order to stay functional. For example, a sophisticated rescue robotic system must be able to change its shape when the space on the path varies with the harsh and unpredictable environment. Space and deep sea explorations are also examples of such variable environments. In most, if not all, engineered systems, the physical components are designed for only limited and predetermined purposes and operation ranges, beyond which the system behaviors are not predictable.

Multi-agent systems are potential solutions to complex tasks. The flexible relationships among the agents provide the system adaptability to deal with changing functional requirements and operation conditions. In order to guide a multi-agent system to fulfill a task, the global task information needs

*Design Computing and Cognition DCC'16*. J.S. Gero (ed), pp. 401-420. © Springer 2016

to be encoded into local rules for agents to follow. The adaptability of the system depends on the encoding process and the range of applicability of the local rules. Most of current approaches encode the task information into agent behaviors. In order to generate suitable rules for given tasks, designers must either specify rules based on their knowledge, or provide algorithms like genetic algorithm to optimize the parameterized rules. In both cases, external supervision is needed, limiting the system adaptability.

Our previous field based behavior regulation approach to cellular self-organizing (CSO) systems [1][2] separates the task encoding process from the system model design process by translating the tasks and environment information into a task field, and the agents operate in the task field by moving from the "higher place" to the "lower place" in the field. The current approach, however, has several limitations, making it difficult to deal with system formation problems. First, in the earlier field based approach [1], each agent operates independently based on its own sensed field information. The system formation is largely an emergent result. Although the emergence offers adaptability, the lack of explicit relations between the agents makes it hard to generate a specific formation, such as a tube or a ring, when the task demands it. Second, in the later dynamic social structuring approach [3] agents form local social structures based on the predefined social rules. Although the introduction of social rules has made it possible to generate local forms and increase the global productivity, the coding of rules can be domain and designer dependent. Third, the approaches to CSO systems developed thus far generally lack learning capability, making it difficult to deal with complex system formation problems where changes need to be learned by the agents during the process of operation.

In this paper, a self-organizing map (SOM) based approach is proposed for CSO system design. The goal here is to apply unsupervised learning and build a system model which is compatible with the task fields generated by the field based behavior regulation. Each agent in the system is treated as a neuron in a SOM, with the ability to sense the environment and communicate with other agents. Tasks are represented as multiple task fields external to the system, and the cooperation among the agents is represented as the social field of the system. The system has the ability to organize itself to fulfill system formation tasks via matching its social field with the external task fields. In the rest of this paper, recent research related to multi-agent systems, field-based regulation, and shape formation is reviewed in Section 2 together with a short introduction to self-organizing map. The adaptive system formation problem and the framework of our approach are introduced in Sections 3 and 4, respectively. Three case studies are described in Section 5 to demonstrate the effectiveness of our approach. Section 6 draws concluding remarks and points to future research directions.

## Related work

In the past decade, design for adaptability has been popular in the field of engineering design and more efforts have been made to investigate multi-agent system design. A multi-agent system consists of multiple robotic agents that are supposed to work with each other to fulfill global level tasks based on their local information. The design of multi-agent system is usually a bottom-to-up process, with a focus on developing a suitable interaction model of multiple agents. Two types of approaches are commonly used in building such models.

The structural design approach is initially inspired by social structures and bio structures in nature. Fukuda and Ueyama [4] compared robotic system with social system, and pointed out the importance of system structure for the intelligence of the system. Kawauchi et al. [5] further proposed the "CEBOT" with genetic knowledge production algorithm that achieved self-organization and self-evolution with a "distributed intelligence system".

Also inspired by social structures, we introduced the concept of field based behavior regulation, in which agent behaviors are regulated by both social field and task field [1][2]. The task field is used to represent the task and the environment for the system. The social field is formed by introducing social rules in multi-agent systems [3]. This dynamic structure can enhance the self-organizing functionality for multi-agent systems using both general and context-based social rules among the agents in the system.

Behavior modeling and optimization is another approach to designing agent interaction model. Our previous work proposed a COARM behavioral model [2] that specifies agent behavior as the composition of five primitive behaviors: Cohesion, Avoidance, Alignment, Randomness and Momentum. A genetic algorithm is used to search for the optimal composition for different tasks such as foraging and habitat construction.

Researchers have applied multi-agent systems to solve shape and structure formation problems. Nagpal et al [6] developed an algorithm for multi-agent systems to form arbitrary predefined shapes. A compiler is used to translate global shape information to local agent rules. Bai and Breen [7] proposed a cell aggregation model to achieve self-organizing shape formation. In this model, an artificial scalar field introduced for cells to move in the direction of the field gradient. A genetic Programming method is used to discover the relationship between different field functions and the shapes formed by the cells. Doursat [8] presented a model using non-random genetic rules to achieve self-assembly and pattern formation. In his approach, self-assembly and pattern formation are integrated in loops to form complex

shapes or structures. Werfel [9] proposed a 3-dimensional model to implement low-level primitives shown in biological system. In his model, the gradient of morphogen field is used to guide modular robots to grow into structures of desired sizes, and provide them with position information.

De Rosa et al [10] proposed a shape formation algorithm for lattice-arrayed modular robots. In their algorithm, the desired shape is compiled into a plan which guide the movement, creation and deletion of the void space in the lattice, transforming the lattice into the desired shape. Tolley and Lipson [11] presented an approach for stochastic assembly of modular robots. The modular robots are put in a fluidic tank, and they can self-reconfigure into different 3-dimentional shapes depending on the fluidic environment.

The current multi-agent system formation approaches mostly involve supervised learning such as genetic programming or compiling that requires pre-knowledge of the target forms. To deal with the uncertainty in the real world, unsupervised learning is more preferable since the information about the environment and the needed formation can be unknown and human interventions to the system are not possible. Self-organizing map is an unsupervised artificial neural network developed by Kohoen [12]. It has been successfully applied in the areas including image processing and speech recognition [13][14].

With a high degree of topological order, SOM can be used in multi-agent systems for structure design through organizing the relationships among the agents. Although there is little direct research to apply SOM to multi-agent systems, some work showed how SOM could be used to solve inner-system relationships and physical pattern matching problems. Kiang et al. [15] applied SOM as a clustering tool in group technology, where SOM was used to address the part machine relationships for grouping the parts. Kit [16] et al. developed the Location Aware Self-Organizing Map to discover visual features of geographical locations.

One major barrier for applying SOM in multi-agent systems is to map the codebooks (related to the social field discussed below) to the data space density (related to the task field discussed below). This problem is represented as magnification control in vector quantization. Villmann and Claussen [17] explored different methods to control the magnification in SOM with regard to the typical SOM learning rules, and summarized them into three types. Localized learning algorithms introduce local learning rates and can achieve arbitrary magnification. The other two types of magnification learning algorithms are winner-relaxing learning and concave-convex learning [18][19]. Compared with localized learning, winner-relaxing learning and concave-convex learning have the advantage of independence on the data distribution [17].

## The problem: adaptive system formation

In this research, the problem of adaptive system formation is considered as a kind of design problem, in which a set of task properties or requirements is given and a system is formed to accomplish the task. The difference between this problem and the traditional design problems is that the system to be designed must be able to sense the changing task requirements and environments and adaptively form and configure itself in response to the changes. Therefore the system formation problem in this sense is a meta-design problem: it is about designing a self-design (i.e., formation or configuration) mechanism.

Examples of system formation include shape formation [7], structure and topology self-configuration [20], and more recently programmable matter [21]. Depending on the tasks, various approaches can be applied to solve system formation problems. In many robotic applications, sophisticated *mechanical mechanisms* are developed to facilitate dynamic formation of systems. In an *evolutionary approach*, a system evolves its formation overtime through generations of computational evolution. When system performance measures are difficult to obtain, *unsupervised learning approaches* can be effective for systems to attain their needed formations.

The long term goal of our research is to device mechanisms that can allow systems to adapt to changing tasks and environments by changing and evolving system formation of shapes, structures, and functional components autonomously without human intervention. As the first step toward this goal, in this paper, we apply an unsupervised learning approach, self-organizing map (SOM), to solving shape formation problems.

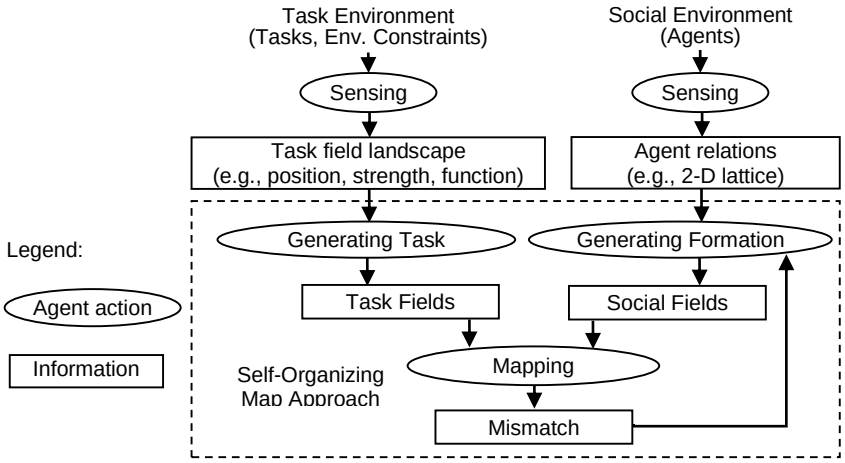



**Fig.1** A proposed system formation approach

The adaptive system formation framework is developed based on our previous work on cellular self-organizing (CSO) systems [1][2][3]. As shown in Figure 1, in this framework, task requirements together with environmental constraints are transformed into one or multiple task fields by the agents. At the same time, the agents sense each other and generate a system formation that is supposed to match the task fields. A SOM algorithm is introduced to do the mapping and update the system formation based on the mismatch. The following section provides details of this approach.

## A SOM based system model

### System and Agents

Following the two dimensional SOM modeling convention, a system *Sys* is defined by $k \times k$ *agents* that form a lattice network topology.

*Definition 1 (System):* $Sys = \begin{bmatrix} Agent\ (1,1) & \cdots & Agent\ (1,k) \\ \vdots & \ddots & \vdots \\ Agent\ (k,1) & \cdots & Agent\ (k,k) \end{bmatrix}$

where $Agent\ (a,b)$*: Agent* positioned in ($a$, $b$) in the network.

In the two-dimension model, the task fields are represented in a standard 2-D space $(x,y)$, $0 \le x \le 1$ and $0 \le y \le 1$, which is also called *working space*. The system formation in this case is against this space. Therefore, each agent, in addition to the network position *(a, b)*, holds an attribute of *position in the task field*: $(x,y)$. Furthermore, in order to differentiate between different task fields that demand different agent functions, we introduce another agent attribute called *function-mode: f*, which can take an integer value to indicate a specific function mode of the agent. Therefore, we have three more agent attributes in addition to ($a$, $b$): $Agent\ (a,b;\ x,y;\ f)$.

### Task field

In the CSO framework, tasks together with environmental situations are represented as "task fields" in which agents seek and move to attractors [1]. In this research, the distribution of "attractors" represents the tasks requirements. The attractors can be singular or can be evenly distributed over an area. We introduce the concept of "task field strength" to capture the attractors. Since there can be multiple tasks involved in a single application, each task should have its own "task field strength distribution." In the special case of two-dimensional task field space, the field strength distributions of two

tasks can be "parallel" meaning that the two tasks can be performed concurrently or "sequential" meaning only one can be performed at a time. We have the following task definitions.

*Definition 2* (*Task Field Strength Distribution*):

$$TF_i(x,y) = tFLD(Env, task_i)$$

where $Env$ is the environment for the system to perform $task_i$, and $tFLD$ is a task field formation operator for generating the field strength distribution of the task.

In case of multiple task fields, the overall task field strength at a given time and position in the field depends on whether the tasks are parallel or sequential. For example, an agent can have the tasks of moving towards a target, and moving away from an obstacle at the same time. Then the overall task field strength distribution can be calculated as:

$$TF_{overall}(x,y) = \sum \omega_i(x,y) * TF_i(x,y)$$

where $\omega_i$ is the weight function for task field *i*.

On the other hand, an agent can have the tasks of searching for food, and carrying food back to the nest. However, the agent can address only one of the two tasks at a time. In such case, the overall task field can be represented by the following equation:

$$TF_{overall}(x,y) = \begin{cases} TF_1(x,y), & Function\ Mode\ 1 \\ TF_2(x,y), & Function\ Mode\ 2 \\ \vdots & \end{cases}$$

In most cases, both of the above two methods are needed to add all task fields together. Thus in a more general form, the overall task field can be represented combining definition 2 and definition 3.

*Definition 3 (Overall Task Field Strength Distribution):*

$$TF_{overall}(x,y) = \begin{bmatrix} \sum \omega_{1i}(x,y) * TF_i(x,y) \\ \sum \omega_{2i}(x,y) * TF_i(x,y) \\ \vdots \\ \sum \omega_{mi}(x,y) * TF_i(x,y) \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1n} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{m1} & \omega_{m2} & \dots & \omega_{mn} \end{bmatrix} \begin{bmatrix} TF_1 \\ TF_2 \\ \vdots \\ TF_n \end{bmatrix}$$

where *n* is the number of all the tasks for the system, and *m* is the number of the function modes for each agent.

**Social field**

Solving the adaptive system formation problem with a CSO systems approach requires the agents in the system to work together to update their formation in response to the needs and changes of the tasks. As shown in Figure 1, a specific system formation in our approach is manifested by its corresponding social field. Similar to the task field, a social field can be characterized by its "social field strength" and "social field strength distribution" over the task field space $(x, y)$, $0 \leq x \leq 1$ and $0 \leq y \leq 1$. As Figure 1 indicates, the mapping between the social field and the task fields provides guidance for the agents of the system to adjust their formation in order to finally achieve minimum mismatch with the task fields.

Our previous work has shown that social structuring among individual agents plays a very important role for self-organizing systems to achieve desired performance [3]. In this research, the social structuring is achieved in two ways (see Figure 1). First, the "Agent Relations" are predefined between the agents. As shown in *Definition 1*, a 2-dimension lattice network topology is employed in this paper for modeling agent relations. Each agent ($a$, $b$) has 2 to 4 connections depending on whether ($a$, $b$) is located in the corner, edge or center part of the network. These connections form the neighborhood of agents. During the process of system formation and update, neighbors influence each other to maintain a desirable formation for given tasks. Second, a *social field strength operator* is introduced for agents to assign and evaluate the social field strength at any given position ($x$, $y$) in the task field space. Based on the SOM algorithm, an ideal system formation is the one that has the compatible "social field strength distribution" with the given "task field strength distribution." Mismatch between the two leads to agents in the system working with their neighbors to adjust their system formation toward better ones. We introduce the following definitions.

*Definition 4 (Social Field Strength Distribution for function model j):*

$$SF_j(x, y, t) = sFLD(Sys, Function\ Mode\ j)$$

where $Sys$ represents the system network, $sFLD$ is a field formation operator which transforms the information from $Sys$ to social field strength under *Function Mode j* at point ($x$, $y$).

*Definition 5 (Overall Social Field Strength Distribution):*

$$SF_{overall}(x, y) = \begin{bmatrix} SF_1(x, y) \\ SF_2(x, y) \\ \vdots \\ SF_m(x, y) \end{bmatrix}$$

where $m$ is the number of all the possible *Function Modes* of each *agent* in the system.

**Matching social and task fields**

The ideal system formation for a given *task j* is determined by assessing the *level of match* between task field strength and the social field strength. The ideal match is that the two strengths are equal. Otherwise, the system formation can be "under-matched" or "over-matched" for the given task. We introduce the following definition:

*Definition 6 (Level of Match for Task j):*

$$Match_j(x, y) = SF_j(x, y)/TF_j(x, y)$$

where $SF_j$ and $TF_j$ are the social field strength and task field strength for task j at point $(x, y)$.

*Definition 7 (Overall Level of Match):*

$$Match_{Overall}(x, y) = \begin{bmatrix} Match_1(x, y) \\ Match_2(x, y) \\ \vdots \\ Match_m(x, y) \end{bmatrix}$$

From the above definitions, it can be seen that the best match happens when Match $(x, y) = 1$. When it is "<1", the system is called "under-matched" (indicating the system is not capable enough to fulfill the task) and when it is ">1" the system is called "over-matched" (the system is overly capable for the task, indicating waste of agent resources).

**SOM based system formation algorithm**

The approach proposed in this paper aims to employ an unsupervised learning to generate system formations based on the social field and task fields, so that the system does not need to go through iterations to find the parameters in structural or behavioral models for convergence. Self-Organizing Map is one of the most popular unsupervised machine learning algorithms, and the topology of SOM network shows great potential for achieving adaptive system formation. The learning process of our approach includes the following two steps.

**Step 1: Distribution**

This step is to update the position in the task space $(x, y)$ of each agent. In this step, a variant of SOM algorithms is applied, as shown below:

1. Initialization

The system is initialized by deploying the agents into the working space (i.e., task space). In this paper, it is a 2-D space $D^2$. Each $Agent\ (a, b)$ has a vector $(x, y, SF)$. $(a, b)$ represents the topological position of the agent in

the lattice of the SOM. $(x, y)$ represents the position of the agent in the 2D working space. $(x, y)$ is randomly picked during initialization.

*SF* is the overall social field strength at $(x, y)$ and is calculated by the number of agents within the neighborhood of distance c around $(a, b)$ .

2. Competition

A random point $(x_p, y_p)$ is picked up from the working space. The picked point has a vector $(x_p, y_p, TF)$. *TF* is the overall task field strength at $(x_p, y_p)$, which is a vector. All the agents in the system compete against each other to find the one which is closest to $(x_p, y_p)$. The winner is selected as the Best Matching Agent (BMA), denoted by *Agent* $(u, v)$.

3. Cooperation

The BMA then communicates with their neighbors and addresses the task field along with the neighbors. The "amount" of cooperation depends on the distance between their positions in the lattice network.

*Definition 8 (Neighborhood Function for Agent (a,b)):*

$$k_{(a,b)} = exp(-\frac{(u-a)^2 + (v-b)^2}{2 * {p_1}^2})$$

where $(u, v)$ is the position of the BMA in the network, $p_1$ is a parameter which can be adjusted to control the strength of cooperation inside the system.

4. Adaptation

All agents update their vectors according to their neighborhood function and the level of task-social field matching at the picked point $(x_p, y_p)$, indicated as learning rate.

*Definition 9 (Learning Rate at* $(x_p, y_p)$*):*

$$\sigma(x_p, y_p) = exp(-\frac{\frac{|SF(x_p, y_p)|}{|TF(x_p, y_p)|}}{2 * {p_2}^2})$$

where $|SF(x_p, y_p)|$ is the norm of $SF(x_p, y_p)$, $|TF(x_p, y_p)|$ is the norm of $TF(x_p, y_p)$, and $p_2$ is a parameter which can be adjusted to influence of satisfaction has on the distribution of the agents in the system.

Finally, the new position of *Agent (a, b)* in the working space is determined by the following equation:

$$(x_{(a,b)}, y_{(a,b)}) = (x_{(a,b)}, y_{(a,b)}) + p_3 * \kappa_{(a,b)} * \sigma(x_p, y_p) * ((x_p, y_p) - (x_{(a,b)}, y_{(a,b)}))$$

where $p_3$ is a parameter which can be changed to control the rate of learning for the system.

**Step 2: Differentiation**

Noticing that the function mode of each agent is not changed in the distribution step, it is updated in the second step, the differentiation step. In this step, the function modes of agents are updated based on their level of matching as defined in Definition 6 & 7 according to the Function Mode Selection Diagram of Figure 2. In general, an agent chooses the function mode corresponding to the task field that the agent has the best match.

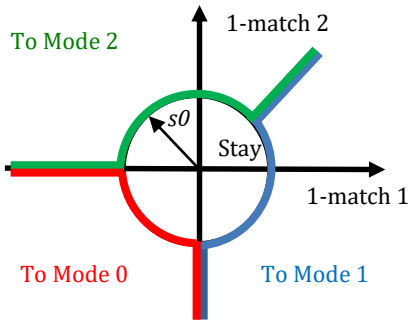



**Fig.2** Function mode selection diagram example

For example, as shown in Figure 2, each agent has three different function modes, with Mode 0 being the default standby mode. When an agent updates its function mode, the position of its matching vector determined its behavior. If the matching vector goes across the blue boundary, the agent switches to Mode 1, so as the green boundary for Mode 2 and red boundary for Mode 0. If the match vector does not intersect any boundary, the agent stays at its old mode. *s0* is a parameter which can be adjusted to control the sensitivity for agents to switch function modes. Randomness maybe added to the function mode changing rule to increase the adaptability of the system.

The process introduced above provides an approach to apply SOM to solving system formation problems. Using this approach, in a real application, the task fields can be manipulated by changing the weight matrix $\omega_{m*n}$, and a detailed agent behavior model should be designed by specifying the parameter $p_1$, $p_2$, $p_3$, the neighborhood distance $c$ and the Function Mode Selection Diagram. The Following section provides specific case studies to show how the process described above can be implemented.

## Case studies

The self-organizing map algorithm is commonly used to organize high-dimensional data and visualize them in low-dimensional maps [16]. The great performance of self-organizing map in vector quantization shows the possibility of building unsupervised algorithms using SOM for adaptive system formation. Several cases simulating shape formation applications are developed to validate this possibility. Matlab was used as the platform for the simulation to take advantage of its capability in calculating vectors and matrices. Figure 3 shows an example of the simulation result.

Figure 3 represents the simulation result for shape formation in a 2D space. The horizontal axis represents the *x* axis; the vertical axis represents the *y* axis. Task fields (not shown in Fig 3) distributes in the area of {(x, y)| 0<x<1, 0<y<1}. Nodes with different color represent agents working under different function modes. The links between agents represent the physical connections between agents. The length of the physical connection can be changed, while the topology of the agents is predefined. Depends on different cases, the nodes and links have different specific meaning, which will be shown in the case studies below.

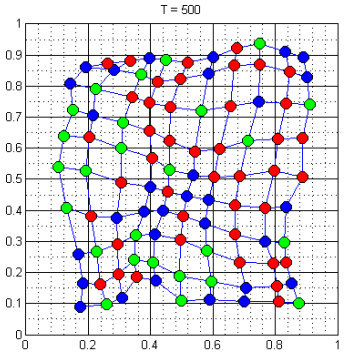
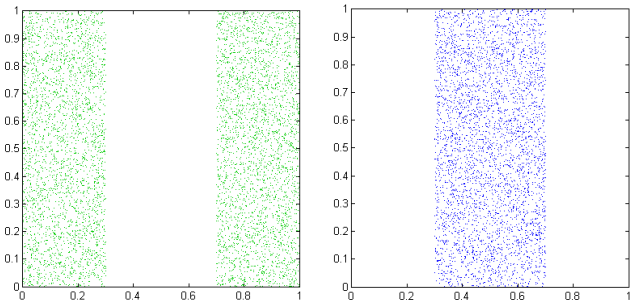



**Fig.3** Example of simulation results

### Case 1: Self-organizing tube formation

Tubes and pipes are used to transfer fluids like air or water. While most tubes are pre-made and then installed in their applications, a self-organizing formation system will be more flexible for installation and repair. In this



(a) TF1 Distribution (b) TF2 Distribution

**Fig.4** Task field distribution

case study, we demonstrate the capability of our approach in using multiple non-overlapping uniform discrete task fields. As shown in Figure 4, the working space is a 2-D space with boundaries from 0 to 1.

Task Field 1: Tube Wall Construction Field TF1. TF1 indicates the positions of the boundary of a tube in 2-D. At the boundary, tube wall is needed to isolate the fluid from the environment. TF1 is distributed over the work space with the task field strength equals to 1 at green areas in Figure 4(a).

Task Field 2: Flow Resistance Demand Field. TF2 indicates the resistance needed to control the flow rate at each point. TF2 is distributed over the work space with the task field strength of 1 at blue areas in Figure 4(b).

The specific parameters used in the simulation are shown in Table 1.

**Table 1** System parameters

| Size of Network | Simulation Steps *T* | Parameter *p1* | Parameter p2 | Parameter *p3* | Neighborhood distance *c* | Switch Threshold *r0* |
|---|---|---|---|---|---|---|
| 10*10 | 500 | 3-0.05 | 0 | 1-0.1 | 0.2 | Rand(1,1)/2 |

Each agent in the system is programmed to have three function modes with regard to the two task fields as in Table 2.

**Table 2** Function modes

| Function Mode | Color | Size & Shape | Function | Related Task Field |
|---|---|---|---|---|
| 0 (Default) | Red | Small, Circle | Cause negligible resistance for the flow | None |
| 1 | Green | Long, Block | Seal the flow with other agents in this mode | TF1 |
| 2 | Blue | Large, Circle | Cause certain resistance for the flow | TF2 |

## Results

The simulation results with varying strength of each task field are plotted in Figures 6 (a) through (f). Here we make the initial task field strength fixed as 1, and adjust the weight operator $w$ for TF1 and TF2 to change the corresponding task field strength for each simulation. Figure 6 shows simulation results for different $w_1$, and $w_2$. Figure 6(a) shows an initial distribution of the system: all the agents are randomly distributed in Mode 0.

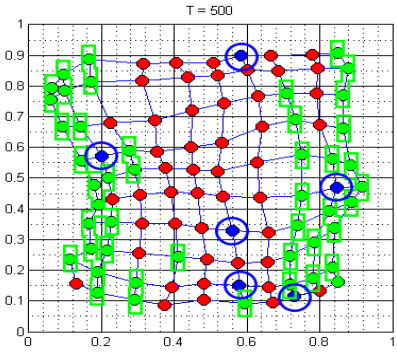
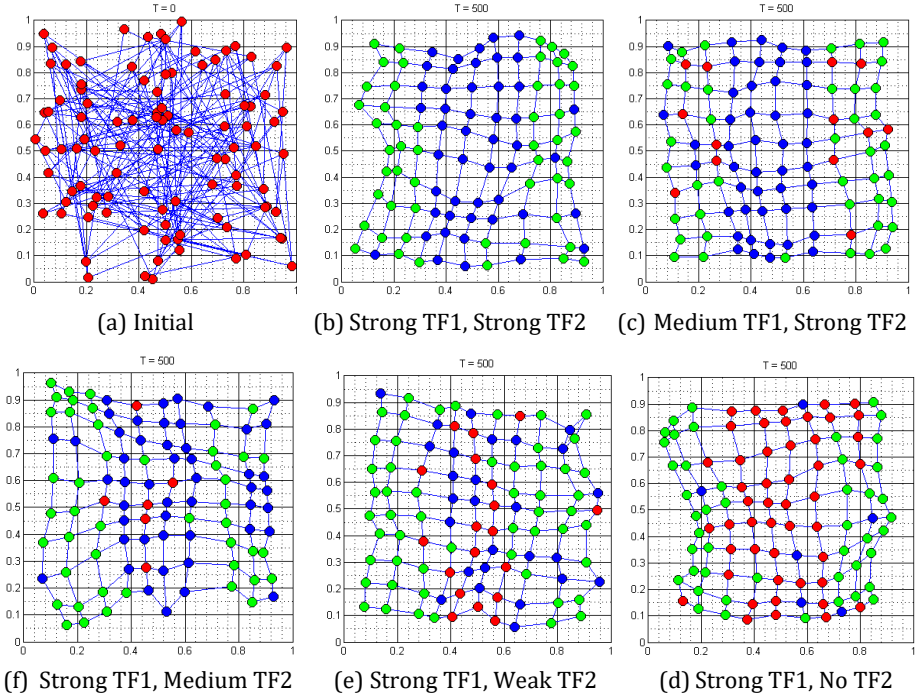



**Fig.5** Function modes

Figure 6(b) shows the result of a simulation with both task fields are strong ($w_1$=$w_2$=15), which means a thick tube wall and high resistance to the flow are needed. The results shows 52 agents are in Mode 2, forming the wall, and 42 agents are in Mode 1, creating large resistance inside the tube.

Figure 6(c) shows the result of a simulation with a medium TF1 ($w_1$=10), and a large TF2 ($w_2$=15), meaning a medium tube wall and high resistance are needed. In this simulation, 38 agents are in mode 2, forming a thinner wall. 44 agents are in Mode 1 inside the tube, which is similar to Figure 6(b).




(a) Initial (b) Strong TF1, Strong TF2 (c) Medium TF1, Strong TF2

(f) Strong TF1, Medium TF2 (e) Strong TF1, Weak TF2 (d) Strong TF1, No TF2

**Fig.6** Self-organizing tube simulation results

Figure 6(b) and Figure 6(d) to Figure 6(f) shows the simulation results of decreasing TF2 ($w_2$=15, 10, 5, 0 respectively) with the same strong TF1 ($w_1$=15). As can be seen from these figures, when TF2 decreases, fewer agents inside the tube is in Mode 1 (42, 34, 17, 4 respectively), which means the resistance inside the tube is decreasing, correctly responding to the changing task demands.

### Case 2: Self-organizing structure formation

The first case study validates that the proposed approach can be used in applications where task fields are “tiled” to each other. In addition, it showed that task field strength can be used to control the formation of the system. In

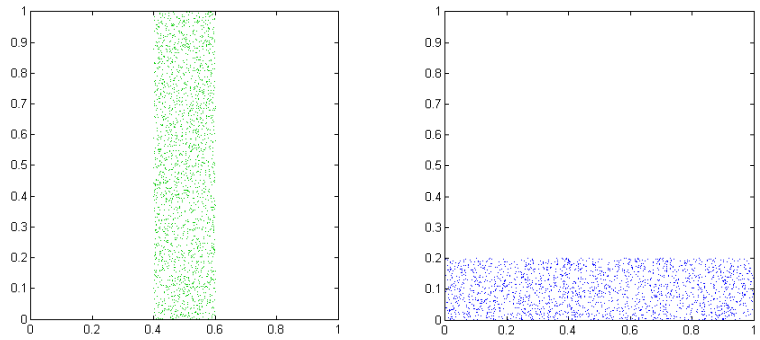
this case study we show that the approach is also applicable for multiple intersecting task fields.

The task is for the system to form a multi-member structure, each member requires different materials/components for achieving different functions. Two task fields are used to indicate required formation of the structure as shown in Figure 7.

Task Field 1: Compression Field TF1 shows the expected path for the weight support of a structure. TF1 is distributed over the work space with the task field strength equals to 1 at every point.

Task Field 2: Tension Field TF2 shows the expected path for the tension force which will be loaded on the structure. TF2 is distributed over the work space with the task field strength equals to 1 at every point.

The same system parameters are used as in Case 1, and each agent in the system is programmed to have three function modes regarding to the two task fields as in Table 3.



(a) Compress Field Distribution (b) Tension Field Distribution

**Fig.7** Example of task field distribution

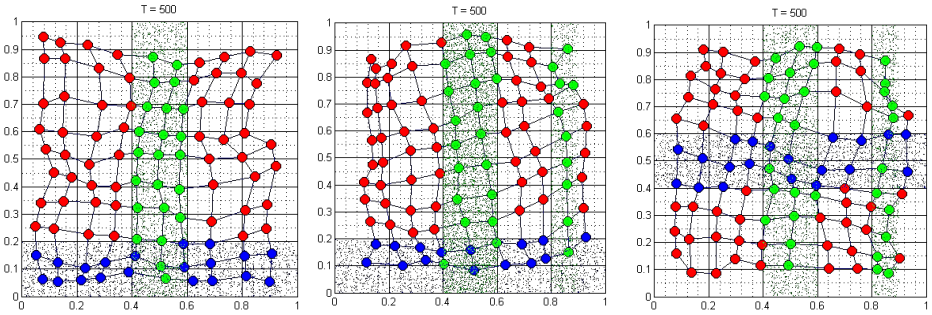
**Table 3** Function modes

| Function Mode | Color | Connection | Function | Related Task Field |
|---|---|---|---|---|
| 0 (Default) | Red | None | Idle and ready to be applied | None |
| 1 | Green | Compression force | Support weight (e.g., a beam) | TF1 |
| 2 | Blue | Tension force | Maintain leveling (e.g., a slab or beam) | TF2 |

### Results

By changing the task field distribution, we can control the formation of the structure. The simulation results are shown below in Figure 8. From Figure 8, we can see that the proposed approach can give accurate results to map

the agent function modes to the task fields. The distribution of green and blue agents can adequately adapt to the changing arrangement of the task fields, forming needed structure to fulfill the tasks.




(a) TF1=15 (V), TF2=15 (H) (b) TF1= 15(V), TF2=15(H) (c) TF1=15(V), TF2=15(H)

**Fig.8** Simulation results (V=Vertical, H=Horizontal)

### Case 3: Smart material formation

The two cases described above have evenly distributed task fields within the defined geometry boundary. The agents going through the self-organizing mapping process can mostly form proper shapes and structures to fulfill the designated tasks. In many other applications, however, the task distribution may not be even, and certain gradient of change of strength along a given direction may be needed for certain system formation purpose. Forming smart materials by depositing multiple kinds of materials in different densities and mixes along given directions and depth is an example of such applications. Two task fields are used in this case study.

Task Field 1: Thermal Field TF1 has its strength either fixed (1 or 8) or decreasing along the direction of $x$ (15*(1-$x$) or 8*(1-$x$)), shown in Fig. 9.

Task Field 2: Flexibility Field TF2 always has its strength increasing along the direction of $x$ (15$x$ or 8*$x$), as shown in Figure 9.

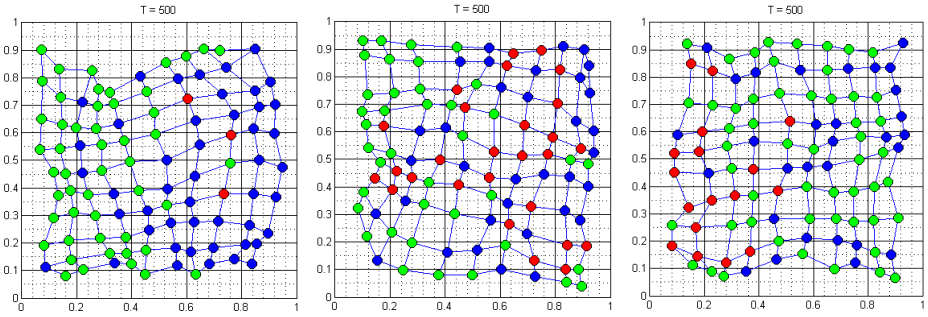
The same system parameters are used as in Case 1, and each agent in the system is programmed to have three function modes regarding to the two task fields as in Table 4.

**Table 4** Function modes

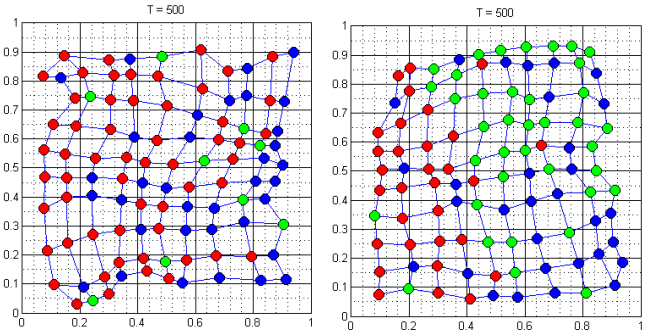
| Function Mode | Color | Connection | Function | Related Task Field |
|---|---|---|---|---|
| 0 (Default) | Red | None | Idle and ready to be applied | None |
| 1 | Green | Thermal effect | Resist heat | TF1 |
| 2 | Blue | Flexibility | Allow bending and torsion | TF2 |

## Results

In each of the simulation, the two task fields TF1 and TF2 cover the entire 2D working space and are completely overlapped. Figures 10 (a) through (e) show the system formation results of various combinations of the distributions of TF1 and TF2. From the figures it can be seen that the distribution of green "heat resistors" and blue "flexiblers" are mostly consistent with the demands of the two task fields TF1 and TF2, respectively. Because the working space is only 2-dimentional, it is hard to realize how such distribution can lead to smart materials. When the working space is 3-D or *n*-dimensional the physical realization of such distributions becomes imaginable.




(a) TF1=15·(1-*x*), TF2=15·*x* (b) TF1= 8·(1-*x*),TF2= 8·*x* (c) TF1= 8, TF2= 8·*x*




(d) TF1= 1,TF2= 8·*x* (e) TF1= 8·*y*, TF2= 8·*x*

**Fig.9** Overlapping task fields simulation results

The comparison of the simulation results demonstrates two important features of our system:

- The distributions of agents in different functional modes are properly mapped to the distributions of multiple task fields.
- The number of agents activated in different functional modes is determined by the varying strengths of multiple task fields.

However, the results also show that even though the task field densities are varying linearly, the agent distributions are not linear. The distortion of the results is possibly because of the nonlinearity of SOM, the randomness of function mode selection, the border effect, and the limited number of agents. Our future research will address these issues.

## Concluding remarks and future work

Solving adaptive system formation problems is essential for developing self-organizing engineered systems. Many natural systems, including both physical and biological systems, possess the capability of dynamically evolve or develop forms, structures and functional components in response to the environmental changes and survival needs. Taking advantages of the natural field distributions, such as gravity and morphogen, natural systems can develop, maintain, and change their formations autonomously. The challenge for developing engineered systems in a similar way as nature is to devise general and yet powerful mechanisms (i.e., physical agents) and algorithms that can be applied to solve real engineering problems. Recent progress in micro robotics as well as drones has offered mechanism opportunities, whereas challenges remain with seeking proper self-formation algorithms.

In this paper, a SOM based approach is proposed to design CSO systems for solving adaptive system formation problems. As an unsupervised learning algorithm, SOM provides certain level of intelligence and makes systems more flexible for complex tasks. The core idea behind the proposed approach is to transform SOM from an artificial neuron network into an agent network by using field-based behavior regulation. The task fields are used to capture the task requirements, and the social field represents the system formation. The CSO system "redesigns" itself automatically by matching the social field with the task fields. The case studies demonstrated the ability of the SOM-based approach.

Compared with existing research, our approach shows uniqueness in the following aspects:

- The information of the tasks is embedded in task fields, external to the multi-agent system. The structural model and behavior model of our approach is independent from tasks, which means the same system with the same configuration can be used to perform different tasks in different environments. In case of shape formation, the change of the task fields will guide the agents self-organize into different shapes.
- Our approach has the capability of dealing with multiple task fields. The case study results show that agents in the system are able to differentiate

into different "species" according to their tasks. Differentiation is the fundamental capability for biological systems to evolve sophisticated functional organs. Our approach aims toward that direction.

- In our approach, agents do not need to synchronize their coordinates for gaining global spatial information in order to choose their behaviors. Instead, the agents are engaged in a parallel mapping process, which increases the speed of self-organization, but also makes it difficult for the system to perform formation tasks that requires precision.
- From a SOM algorithm perspective, in comparison with existing density tracking network algorithms, our approach is a bottom-up one. The implementation of the algorithm is distributed to individual agents rather than a single computer. By using the field regulation, the system captures multiple 3-dimensional dataset in 2-dimensional space at the same time, making the multiple mappings and calculations more efficient.

The long term goal of our research is to devise algorithms that can allow systems to adapt to changing tasks and environments by changing and evolving system formation of shapes, structures, and functional components autonomously without human intervention. To pursue this goal, we will conduct more case studies with more close-to-real example problems. In doing so, we will further enhance our task field modeling scheme and social field system presentation. Physical implementation is also a future direction.

This paper is based on the work supported in part by the National Science Foundation under Grants No. CMMI-0943997 and No. CMMI-1201107. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

1. Chen, C, Jin, Y (2011) A behavior based approach to cellular self-organizing systems design, *IDETC/CIE 2011*, Aug.28-31, 2011, Washington, DC.
2. Humann, J, Khani, N, and Jin, Y (2014) Evolutionary computational synthesis of self-organizing systems, *AIEDAM*, 28(3) pp. 259-275.
3. Khani, N, Jin, Y (2015) Dynamic structuring in cellular self-organizing systems, *Design Computing and Cognition'14*, Springer, pp. 3-20.
4. Fukuda, T, Ueyama, T (1992) Self-organizing cellular robotic system: Comparison between social system and robotic system, *1992 IEEE Int'l Biomedical Engineering Days*, pp. 39-45.
5. Kawauchi, Y, Inaba, M, Fukuda, T (1992), Self-organizing intelligence for cellular robotic system `CEBOT' with genetic knowledge production algorithm, *1992 IEEE Int'l Conf. on Robotics & Automation*, pp.813-18.

6. Nagpal, R (2006) Self-organizing shape and pattern: From cells to robots," *IEEE Intelligent Systems*, **21**(2) pp. 50-53.
7. Bai, L, Eyiyurekli, M, Breen, DE (2008), Self-organizing primitives for automated shape composition, *IEEE Int'l Conf. on Shape Modeling & Applications*, pp.147-154, Stony Brook, NY, 2008
8. Doursat, R., 2008, The self-made puzzle: integrating self-assembly and pattern formation under nonrandom genetic regulation, *InterJournal Complex Systems*, 2292
9. Werfel, J (2010), Biologically realistic primitives for engineered morphogenesis, *Swarm Intelligence*, Vol.6234 of the series Lecture Notes in Computer Science, pp 131-142.
10. De Rosa, M, Goldstein, S, Lee, P (2006), Scalable shape sculpting via hole motion: motion planning in lattice-constrained modular robots, *2006 IEEE Int'l Conf. on Robotics and Automation*, pp.1462-1468.
11. Tolley, MT, Lipson, H (2010), Fluidic manipulation for scalable stochastic 3D assembly of modular robots, *2010 IEEE Int'l Conf. on Robotics and Automation*, pp. 2473-2478.
12. Kohonen, T (1990), The self-organizing map, *Proceedings of the IEEE*, 78(9) pp. 1464-1480.
13. Galda, H, Murao, H, Tamaki, H (2004), Dermoscopic image segmentation by a self-organizing map and fuzzy genetic clustering, *IEICE Transactions on Information and Systems*, E87-D (9) pp. 2195-2203.
14. Zhang, Y, Tang, Y, Fang, B (2014) Real-time object tracking in video pictures based on self-organizing map and image segmentation," *2014 7th IEEE Joint Int'l Information Technology & AI Conf.*, pp. 559-563.
15. Kiang, MY, Kulkarni, UR, and Kar, YT (1995), Self-organizing map network as an interactive clustering tool-an application to group technology, *2nd Workshop on Info. Tech. and Systems* (*WITS'92*), pp.351-74.
16. Kit, D, Kong, Y, Fu, Y (2014) LASOM: Location aware self-organizing map for discovering similar and unique visual features of geographical locations, *2014 Int'l Joint Conf. on Neural Networks*, pp. 263-270.
17. Villmann, T, and Claussen, JC (2006), Magnification control in self-organizing maps and neural gas, *Neural Computation*, 18(2) pp. 446-469.
18. Claussen, JC, Villmann, T (2005) Magnification control in winner relaxing neural gas, *Neurocomputing*, 63, pp. 125-137.
19. Zheng, Y, Greenleaf, JF (1996) Effect of concave and convex weight adjustments on self-organizing maps, *IEEE Neural Networks*, 7(1) pp.87-96.
20. Ferguson, SM, Lewis, K (2006) Effective development of reconfigurable systems using linear state-feedback control, *AIAA Journal*,44(4) pp.868-78.
21. Derakhshandeh, Z, Gmyr, R, Strothmann, T (2015) Leader election and shape formation with self-organizing programmable matter, *21st Int'l Conf. on DNA and Molecular Programming*, pp.117-132.