Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2014), *28*, 259–275.
© Cambridge University Press 2014 0890-0604/14 $25.00
doi:10.1017/S0890060414000213

Evolutionary computational synthesis of self-organizing systems

JAMES HUMANN, NEWSHA KHANI, AND YAN JIN
Department of Aerospace and Mechanical Engineering, University of Southern California, Los Angeles, California, USA

(RECEIVED April 9, 2013; ACCEPTED February 20, 2014)

Abstract

A computational approach for the design of self-organizing systems is proposed that employs a genetic algorithm to efficiently explore the vast space of possible configurations of a given system description. To generate the description of the system, a two-field based model is proposed in which agents are assigned parameterized responses to two "fields," a task field encompassing environmental features and task objects, and a social field arising from agent interactions. The aggregate effect of these two fields, sensed by agents individually, governs the behavior of each agent, while the system-level behavior emerges from the actions of and interactions among the agents. Task requirements together with performance preferences are used to compose system fitness functions for evolving functional and efficient self-organizing mechanisms. Case studies on the evolutionary synthesis of self-organizing systems are presented and discussed. These case studies focus on achieving system-level behavior with minimal explicit coordination among agents. Agents were able to collectively display flocking, exploration, and foraging through self-organization. The proposed two-field model was able to capture important features of self-organizing systems, and the genetic algorithm was able to generate self-organizing mechanisms by which agents could form task-based structures to fulfill functional requirements.

Keywords: Agent-Based Approach; Computational Design; Design Synthesis; Genetic Algorithm; Self-Organization

1. INTRODUCTION

Engineered systems are becoming increasingly complex because of the rising expectations of customers, the increasing specialization of engineering knowledge, and a push toward deployment in hostile environments. In this paper, we focus on the desire for adaptive systems, which display features such as robustness, flexibility, and resilience. These types of adaptive capabilities are necessary in systems that are faced with changing task requirements or unknown environments. For example, a system designed for lunar exploration can be very different from a robot designed to repeatedly assemble body panels of an automobile. The latter is assumed to exist under supervision in a well-regulated factory, and its inputs can be prescribed, but the former may need to act autonomously when communication with its home base is not feasible, and it must be able to overcome many unpredictable obstacles.

By Ashby's (1958) law of requisite variety, adaptability can only be found in systems that have many possible states. Thus, adaptability is born of complexity, but traditional techniques for engineering synthesis are not well suited for the design of complex systems. Self-organization has been suggested as a strategy for the design of complex systems, and in this paper we propose an approach to the synthesis of *cellular self-organizing* (CSO) *systems* that relies on *field-based behavior regulation* (FBR), *multiagent simulation*, and a *genetic algorithm* (GA) to deal with the combinatorial problem of synthesizing agent parameters.

Self-organization is the spontaneous emergence of global structure and order arising from lower-level components with no global control. A self-organizing system is a distributed system composed of agents with no global knowledge. Typically, these agents interact with only their closest neighbors and immediate environment. The interactions among agents lead to the emergence of some distinct pattern or behavior at the system level. Self-organization is evident in many types of systems. Physical, biological, and social systems have been shown to be complex and self-organized. Compared to traditionally engineered systems, self-organizing systems are far more common, but they are not as well understood (Doursat, 2011). Self-organization has been studied extensively as a naturally occurring phenomenon, but less

Reprint requests to: Yan Jin, Department of Aerospace and Mechanical Engineering, University of Southern California, 3650 McClintock Avenue, OHE430, Los Angeles, CA 90089-1453, USA. E-mail: yjin@usc.edu

effort has been focused on researching self-organization as an intentional design mechanism. Self-organization can be used as an approach to developing systems that must perform in dynamic environments or fulfill changing requirements, for example, biomedical applications, extraterrestrial environments (Payton et al., 2001), and military and search and rescue situations. As the mechanical slowly merges with the natural (Kelly, 1994; Bentley, 2001), the ability to purchase a swarm of small, cheap, insectlike robots with some of the capabilities envisioned by many self-organization researchers (Rubenstein et al., 2012) is becoming more likely, and there is a greater need for understanding how to develop proper behavioral rules for them.

In our previous work, CSO systems have been proposed as a way to embed self-organization into engineered systems to solve tasks adaptively (Zouein et al., 2010; Chen & Jin, 2011). The aim was to design systems that show adaptability: flexibility in the face of changing design requirements, robustness when encountering unfamiliar environments, and resilience to the failure or malfunction of a portion of the system. The agents of a CSO system are simple robots called *mechanical cells* (mCells). Various systems have been described as "cellular" in the literature. The cells can be stationary, as in cellular automata, or mobile, as in Beni (1988) and Bai and Bree (2012). The metaphor is used to convey the idea that these agents are simple and mostly interchangeable, with their real value arising from their interactions and self-organized ability to work together. The mCells in our research are mobile and able to sense each other and their environment within a local radius of detection. They have the ability to make decisions with simple calculations. The mCells do not have a unique identification and do not send messages to individuals, but communicate (if they communicate at all) via one-to-many signaling.

Each mCell has a set of instructions governing its response to environmental stimuli. Zouein et al. (2010) demonstrated a self-organized shape reconfiguration using the CSO assumptions and a "design DNA" (dDNA). A dDNA is composed of common information shared by mCells and a set of behavioral instructions for the cells to follow. Inspired by gene regulation and dynamical systems, Chen and Jin (2011) introduced an FBR strategy for the control of such systems. Rather than explicitly interact with one another, mCells calculate a field of local environmental influences. The behavior of the mCell will then depend on which areas of the field attract it, based on the location's field value. The generality of dDNA and FBR can be exploited to create a diverse range of systems (Chiang, 2012). The current CSO framework, however, relies on a set of ad hoc methods to develop self-organizing mechanisms. To further advance the research, there is a strong need for a systematic and computational approach for CSO system synthesis.

GAs are nature-inspired stochastic optimization algorithms (Goldberg, 1989; Holland, 1992) that operate on a population of solution candidates. Candidates are described by a genome, which is usually a binary string. GAs use the operators of selection, crossover, and mutation to "evolve" the population through many generations until it produces suitable candidate genomes. GAs can efficiently search large optimization spaces, and their population-based approach can keep them from getting trapped in local optima. Because the CSO systems are composed of mCells characterized by dDNA, the GA provides a natural way to evolve and synthesize CSO systems. In this paper we present our evolutionary computational approach to the synthesis of CSO systems.

In the rest of the paper, we first review the related work in Section 2 and then introduce a two-field (i.e., task field and social field) model of CSO systems in Section 3. Based on this model, in Section 4, a GA-based evolutionary computing approach is developed to support the synthesis of CSO systems. Three simulation-based case studies are presented and discussed extensively in Section 5. Section 6 draws conclusions and points to future research directions.

2. RELATED WORK

The computational design synthesis process described in this paper draws inspiration from other studies on artificial self-organizing systems, automated design, and the evolutionary optimization of complex systems.

2.1. Artificial self-organizing systems

Artificial self-organizing systems are systems built by man that display self-organizing behavior similar to natural systems. Artificial self-organizing systems can be built for research, practical purposes, or simply enjoyment. Various systems such as robot swarms can be described as self-organizing systems, and some examples are reviewed below.

Werfel (2012) demonstrated a promising application for CSO systems: collective construction. Drawing inspiration from social insects such as termites, which can build mounds up to 8 m tall (Korb 2011), Werfel developed a system of swarm robots that can build a prespecified two-dimensional shape out of square bricks. Localization and gripping are the main barriers to this type of construction, but Werfel introduced simple remedies for both problems. Claytronics (Goldstein & Mowry, 2004) is an attempt to create millions of tiny interacting robots with no moving parts that can use one another as anchor points for locomotion and ultimately become a new, three-dimensional medium for communication. Beckers et al. (1994) describe a robotic gathering system, where robots move around an arena collecting pucks. Robots are more likely to drop pucks in an area of high puck density. This causes a positive feedback loop that results in a single group of all available pucks.

In all of these cases, the agents fulfilling the task had very little knowledge of the global-level functional requirement. Communication with near neighbors was required in Claytronics, but only indirect communication via the state of the task completion was used in the gathering and collective construction tasks. The agents used were very simple, but with the proper selection of local interaction rules, useful emergent behavior was demonstrated.

2.2. Evolutionary optimization of complex systems

Computer optimization of complex problems is an active area of research in mathematics, computer science, and artificial intelligence. Optimization of such systems is difficult, because the link between the input parameters and the global measured behavior is often unclear and nonlinear. In addition, the behavior of interest may be transient, emergent, or otherwise hard to measure (Calvez & Hutzler, 2006). Nonetheless, there are many successful demonstrations of evolutionary optimization of complex systems in the literature.

Ueyama et al. (1992) used a GA to optimize a path-planning algorithm in a simulation of their distributed CEBOT robotic swarm. Stonedahl and Wilensky (2010) used a GA to develop flocking rules that would exhibit emergent behavior such as cohesive flocking, V-formation, and volatility. GA has been used to tune the update rules of cellular automata in order to complete the "majority classification" task (Mitchell et al., 1994; Crutchfield et al., 1996). The GA evolved intelligent update rules that allowed cellular automata cells to distribute information about local densities to other remote cells. GA's close relative genetic programming was used in (Van Berkel et al., 2012) to combine functional primitives to recreate natural collective phenomena such as firefly synchronization and ant foraging.

These examples show that, given a proper fitness function, evolutionary approaches are a viable strategy for the optimization of complex and self-organizing systems. By application of GAs, genetic programming, or other approaches, researchers have been able to efficiently explore vast search spaces and optimize systems that display complex behavior and require the simultaneous tuning of many input variables.

2.3. Evolutionary and computational design

Automated approaches to traditional design can be aided by processes that build up system functions from component primitive functions. Given a set of primitives, and a grammar for combining them, some of the design work can be done by computers (Sridharan & Campbell, 2005), and GAs have been used to automatically develop function–structure diagrams for routine designs (Jin & Li, 2007). These approaches have found success in the computational synthesis of simple designs. The complex interactions among the agents of self-organizing systems make a function–structure approach to their design very difficult to implement.

Our evolutionary optimization of CSO systems is most similar to the concept of evolutionary embryogeny, which has been used to evolve growth rules for vertical structures that must support a horizontal load (Yogev et al., 2008). In our approach, and in the embryogeny approach, the description of the system goes through a mapping process before the structure of the system is determined. This mapping process is the self-organized task completion in our CSO systems and the growth process in artificial embryogeny. This indirect mapping allows more complex structures to arise from simpler descriptions, which can shorten the description required and shrink the space of the search for a proper description (Bentley, 1999). Unlike artificial embryogeny, we do not allow the cells of our system to grow and divide. Another subtle but important difference is that in artificial embryogeny, the self-organizing rules are executed at design time, specifying the final geometric structure of the system, whereas in our CSO systems, the rules themselves are the product of the design, and the system is allowed to self-assemble at run time.

In our research, we apply GA to evolve the self-organization mechanisms. The challenge is to develop a generic self-organization parametric model that is general enough to characterize the rich self-organizing behavior of agents, and constrained so that it will work within the GA framework for computational synthesis. We introduce a two-field based model below.

3. A TWO-FIELD MODEL OF SELF-ORGANIZATION IN CSO SYSTEMS

A field in general is a mathematical abstraction of influence acting in space. Physical fields such as gravitational, electric, or magnetic fields are ubiquitous, and biology also considers the influence of fields of chemicals. Variations in the concentrations of certain chemicals called morphogens create a field that governs the development of living organisms (Haken, 1978), and mobile cells can follow chemical gradients in search of agreeable environments in a process called chemotaxis. Drawing on the natural idea of fields, we introduce two fields to govern self-organizing behavior.

3.1. Task field

The *task field* of a CSO system represents the agents' perception of fixtures of the environment and objects involved in the system's tasks. Here we include terrain, laws of nature, responses to obstacles, and attraction of the system's goal (when applicable) as elements giving rise to the task field. In the multiagent box-pushing example (Chen & Jin, 2011), the mCells do not communicate in any substantial way, but act entirely based on the task field caused by attracting and repelling forces in the environment. For an agent i, and m objects to be approached and n objects to be avoided in the field, the task field seen by the agent i can be expressed as

$$tField_i = \mathrm{FLD_T}(\theta_1, \theta_2, \ldots, \theta_m, \beta_1, \beta_2, \ldots, \beta_n), \qquad (1)$$

where $\mathrm{FLD_T}$ is a task field generation operator.

In some self-organizing systems, the agents interact with the object of the tasks in such a way that updating its state gives agents information on how to further update its state toward the final goal; this is called *stigmergy* and is a way of using a changing task field to generate emergent behavior. Stigmergy, or work-creating-work, can be thought of as a dynamic task field. Stigmergy in collective robotics can be used for gathering tasks (Beckers et al., 1994; Song et al., 2012) and has been shown to be the mechanism by which termites organize to build their mounds (Korb, 2011). "Extended stigmergy" (Werfel & Nagpal, 2006) is a more extreme example, where blocks in a building task are encoded with radiofrequency identification chips or other mechanisms for conveying more precise messages from agent to agent. In each case of stigmergic self-organization, the interactions among agents were indirect and unsynchronized, meaning that there was a possible delay between the time a message was sent and the time it was received. Because the medium of signaling was also the task object of interest, we can classify this self-organizing behavior as a response to a changing task field.

3.2. Social field

In addition to task situations, an agent's behavior can be affected by other agents. The *social field* arises from agents' influence on one another and forms another layer of information for the system to use. Explicitly distinguishing between task field and social field allows us to design needed social structures for agents to complete their tasks in variable task environments. To cope with a given task field, an agent can explore and develop its social relations (i.e., agent–agent relations), which should resolve possible conflicts and foster cooperative interactions.

Researchers have used various communication strategies among agents to explore social fields. In some systems (Reynolds, 1987; Cucker & Smale, 2007; Chiang & Jin, 2011), agents sense one another's presence and react directly while traveling mostly through empty space. Thus they have little concern with a task field, and the social field dominates. Signals can also be used to build a social field. Communication strategies such as "pherobots" (Payton et al., 2001) or hormone-inspired robotics (Shen et al., 2002) can be described as different ways of creating a social field. An extreme example of a social field is given in Bai and Bree (2012). These authors use genetic programming to evolve mathematical formulas for field generation. Each agent in their simulation propagates a field in its local vicinity according to a complicated mathematical function, and senses the fields generated by others. Agents then follow the gradients of the field to form shapes.

In our research, an agent constructs its social field based on its relations with observable neighboring agents. For agent *i* with *n* observable neighboring agents, we have

$$sField_i = \mathrm{FLD_s}(\varphi_{i1}, \varphi_{i2}, \ldots, \varphi_{in}), \tag{2}$$

where $\mathrm{FLD_s}$ is a social field generation operator and $\varphi_{i1}, \varphi_{i2}, \ldots, \varphi_{in}$ are agent *i*'s relations with other agents. If the social relations can be captured simply by attraction, repulsion, and state information, then we have

$$\varphi_{ij} = \left\{ \begin{matrix} \alpha_j \\ \rho_j \\ \sigma_j \end{matrix} \right\}, \tag{3}$$

where α, ρ, and σ represent a neighbor's attraction, repulsion, and state, respectively.

In self-organizing systems, communication among agents is often not one-to-one. Rather, agents create a field of influence in their vicinity. The omission of one-to-one messaging makes it possible to have a truly distributed and decentralized system, where an individual agent does not even have a unique identification. This adds resilience to the system, because the failure of one agent does not immediately imply failure of the system; another identical agent can always take its place.

With limited sensory capabilities, no complete and static field function can be generated for self-organizing systems, because agents' perception of field values changes with their position (e.g., if a repelling object moves into or out of an agent's sensory range). This makes mathematical proofs of stability difficult, so simulation is used to give statistical confidence in system performance.

3.3. Cellular (agent) behavior

In a CSO system, the behavior of an individual mCell is defined by a mapping process:

$$b = \{S_E, A_E\} \rightarrow A_N, \tag{4}$$

where S_E and A_E are the mCell's current sensory information and actions, respectively, and A_N is the mCell's next chosen action. The overall behavior of the system (BoS) is then

$$\mathrm{BoS} = \{B_1, B_2, \ldots, B_n\}, \tag{5}$$

where B_i represents the set of all possible behaviors $\{b_1, b_2, \ldots\}$ of mCell *i*. A designer must develop a correct behavioral mapping such that the BoS fulfills the system's functional requirements. An FBR has been proposed (Jin & Chen, 2012) for the single *task field* models. In this research, we consider two fields, *task field* and *social field*. Figure 1 illustrates our field-based cellular behavior regulation in the two-field based model context. Equation (6) is a two-field based FBR for given function requirements (FR), environment (Env), and agents $(a_1, \ldots, a_n)$:

$$\begin{aligned} b_{t+1} = {} & \mathrm{FBR_{BS}}(\mathrm{FBR_{FT}}(\mathrm{FLD_T}(\mathrm{SNS_T}(\mathrm{FR_t}, \mathrm{Env_t})) \\ & \times \mathrm{FLD_S}(\mathrm{SNS_S}(a_{1t}, a_{2t}, \ldots, a_{nt}), \ \mathrm{state}))), \end{aligned} \tag{6}$$

where $\mathrm{SNS_{T/S}}$ is the task/social field sensing operator; $\mathrm{FLD_{T/S}}$ is the task/social field generation operator; $\mathrm{FBR_{FT}}$ is the task to behavior field transformation regulator; $\mathrm{FBR_{BS}}$ is the behavior selection regulator; tField is $\mathrm{FLD_T}$(task information), which is $\mathrm{FLD_T}(\mathrm{SNS_T}(\mathrm{FR_t}, \mathrm{Env_t}))$: task field; sField is $\mathrm{FLD_S}$(social information), which is $\mathrm{FLD_S}(\mathrm{SNS_S}(a_{1t},$

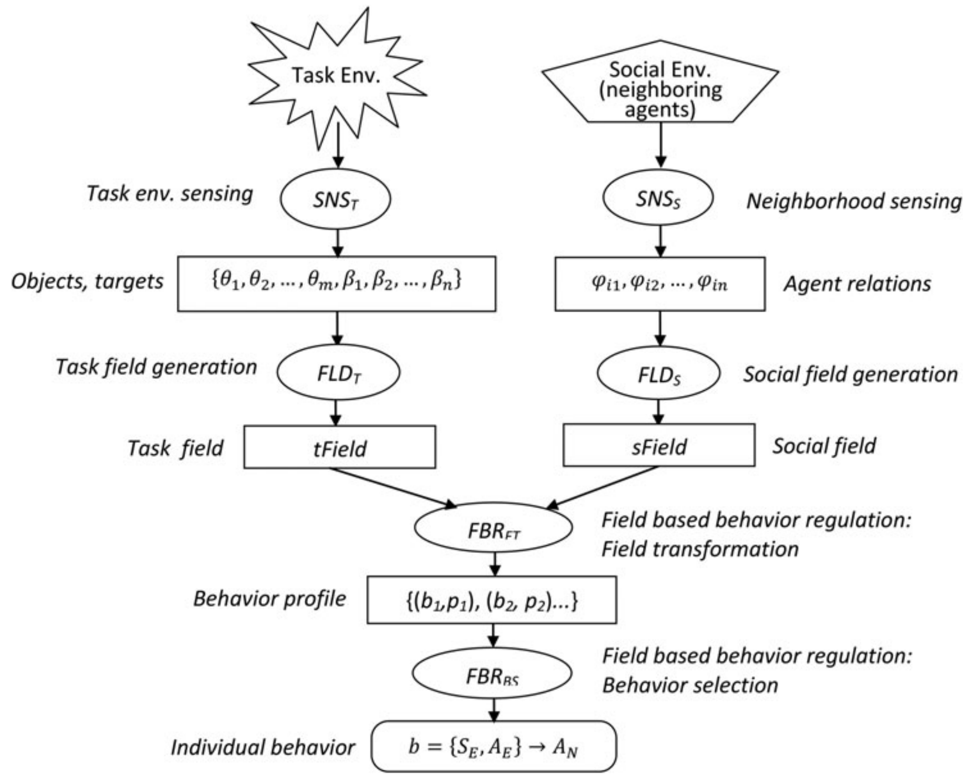

Fig. 1. The two-field based model context field-based behavior regulation .

$a_{2t}, \ldots, a_{nt}$), state): social field; bProfile is FBR_{FT}(tField, sField): behavior profile; and b_{t+1} is FBR_{BS}(bProfile): selected behavior for the action of the next step.

Every mCell makes this decision and acts in parallel in discrete time steps. The behavior output process is a function of the *state* of an mCell and its neighbors, the social field (sField), the task field (tField), and the set of behavioral primitives, where *state* can consist of internal state variables known to a particular agent (e.g., memory, sensor values, and contents) and external variables known to all agents (e.g., location, speed, and color).

SNS_{T} and SNS_{S} are sensing operators that identify an mCell's task environment and social environment, respectively. The task field formation operator FLD_{T} abstracts task and environmental stimuli to create a mathematical task field. At the same time, the mCell can also identify its neighboring agents and form a social field through its social field operator FLD_{S}. Based on the information defined by the two interacting fields, the mCell creates a behavioral profile through the operator FBR_{FT}. The resulting behavioral profile is composed of a set of (*behavior*, *preference*) pairs. The final step is behavior selection based on operator FBR_{BS}.

The sField contains all social relationships affecting an agent. The sField must contain both a topology and a policy. The topology determines the allowable agent relationships (i.e., which relationships have any meaningful effect). The policy determines the strength of interactions between the agents who fulfill one of the allowable relationships. In a general heterogeneous system with M number of agent types, the topology can be described as a matrix of relationships:

$$R = \begin{bmatrix} r_{11} & \cdots & r_{1M} \\ \vdots & \ddots & \vdots \\ r_{M1} & \cdots & r_{MM} \end{bmatrix}, \tag{7}$$

$$r_{ij}^{k} = \left\{\text{trigger}, p\left(SE^{k}, c^{k}\right)\right\}, \tag{8}$$

where r_{ij} is the policy of an agent of type i toward an agent of type j. The trigger is a condition that causes the relationship to take effect. Here, p is a reaction to the external state of neighboring mCell k and its communication, if any. In general, we want these relationships to be anonymous and not determined for any specific mCells, just for mCell types, so r_{ij} should be the same for every agent k of type j. The matrix is not necessarily full or symmetric.

The tField contains important objects, locations, and constraints in the system's task and environment:

$$\begin{aligned} \text{tField} &= \{\text{TO}, \text{TL}, \text{EO}, \text{EL}\} \\ \text{TO} &= \{\text{to}_1, \text{to}_2, \ldots, \text{to}_\text{p}\} \\ \text{TL} &= \{\text{tl}_1, \text{tl}_2, \ldots, \text{tl}_\text{q}\} \\ \text{EO} &= \{\text{eo}_1, \text{eo}_2, \ldots, \text{eo}_\text{r}\} \\ \text{EL} &= \{\text{el}_1, \text{el}_2, \ldots, \text{el}_\text{s}\}, \end{aligned}$$

where TO, TL, EO, and EL are task objects, task locations, environmental objects, and environmental locations, respectively. The address of a goal is an example of a tl_i, and a set of obstacles is an example of the list of environmental objects.

Generally speaking, social field manipulation is associated with a self-organized structure of the system. The fulfillment of the task requirements often depends on the structure of the system. Organization theory claims that organizations form structures that are dependent on their task and environment (Thompson, 1967). This suggests that better system functionality can be achieved through the self-organized emergence of complementary social structure and behavior, and we propose the use of a social field and task field to generate this.

4. EVOLUTIONARY SYNTHESIS OF CSO SYSTEMS

Because self-organizing systems are usually complex, convergence proofs and analysis of them by the application of continuous formulas is nearly impossible, so multiagent simulations are a popular approach to study the behavior of these systems. In addition to applying simulations in a trial-and-error fashion for analyzing system behavior, combining simulations with powerful computational optimization techniques can help generate design directions for complex systems development.

We propose an evolutionary approach to the synthesis of CSO systems to ameliorate the design difficulties inherent in their complexity. Figure 2 illustrates the iterative nature of this approach. The approach corresponds to the traditional view of design in the following ways: *function* is recast as *task* in our model, and *form* becomes a *description*, or *model*, *of the self-organizing system*. Because we do not have analytical tools as powerful as traditional engineering analysis, we do not have a directly analogous component in our model for *behavior*. Instead, we use multiagent simulations to generate the system and simulate its behavior, and a fitness function to evaluate its performance. A GA is employed to iteratively synthesize the description of the system. Figure 2 illustrates an example of how our FBR model can be mapped into a GA environment for evolutionary synthesis.

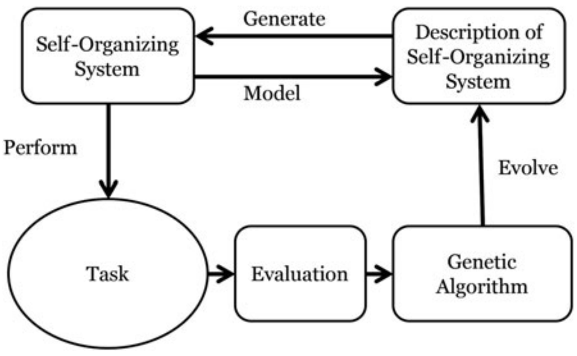

Fig. 2. The model of the cellular self-organizing design process.

For the purposes of this research, we are concerned with the three stages in the right half of Figure 2: namely, the description of the self-organizing system, evaluation of the system's ability to fulfill its task requirements, and the GA that evolves the former according to the latter. We treat the hardware of the agents in the self-organizing system as given. We assume that the task can be simulated using multiagent simulations with enough fidelity to draw conclusions about its emergent behavior. The design effort is then focused on the description, or the model, of the self-organizing system and the GA (together with the evaluation) used to evolve that description.

To achieve desired system behavior, we focus on designing the interactions among agents and reactions to objects in the mCells' task. Narrowing the design focus to interaction rules may still not simplify the design process enough, because the number of interactions among agents can grow nonlinearly with the types and numbers of agents in the system. To deal with the large number of interactions, we apply GA because GAs can efficiently search a large space and have shown encouraging results in similar applications (Ueyama et al., 1992; Trianni, 2008; Stonedahl & Wilensky, 2010; Humann & Jin, 2013). The description of the system should then be parameterized as much as possible to mesh with the GA. If the mCells are designed to respond to their governing fields according to parameterized weights of desired behaviors, as shown in Figure 3, then these parameters become the design variables that the GA acts on, and our approach allows a fast synthesize-simulate loop with an optimization algorithm to be used instead of the traditional synthesize-analyze loop.

5. CASE STUDIES IN COMPUTATIONAL CSO SYNTHESIS

To evaluate our evolutionary approach to the design of CSO systems, we carried out three case studies. The case studies correspond to three different system FRs: flocking, exploration, and foraging. Flocking is the ability for initially dispersed mCells to come together and move as a group. Exploration entails uncovering a certain percentage of the available field in a specified amount of time by sending at least one mCell to each distinct area. Foraging requires finding a food source located far from the home base and returning food from the source back to home. Flocking and exploration use identical sField relationships, and nearly disregard the tField; the mCells move through a space with no distinct tField objects, but the size and shape of the arena do have some effect. Foraging uses more complicated sField relationships and a stronger tField. The foraging agents not only react to one another's position and location but also are allowed to sense and broadcast state information. The tField in foraging consists of a food source and a home beacon. These features can attract or repel mCells according to their state.

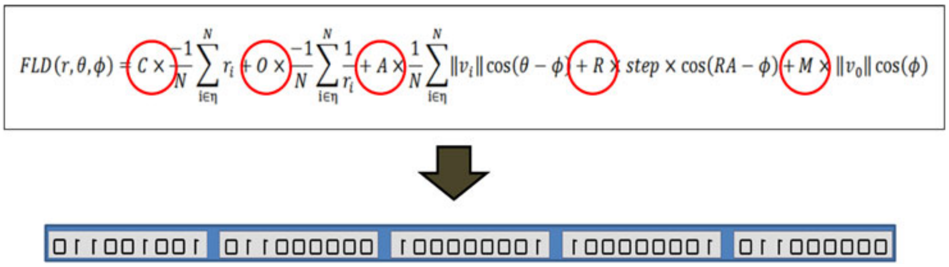
$FLD(r,\theta,\phi) = C \times \frac{-1}{N}\sum_{i\in\eta}^{N} r_i + O \times \frac{-1}{N}\sum_{i\in\eta}^{N}\frac{1}{r_i} + A \times \frac{1}{N}\sum_{i\in\eta}^{N}\|v_i\|\cos(\theta-\phi) + R \times step \times \cos(RA-\phi) + M \times \|v_0\|\cos(\phi)$
01100100
01100000
100000001
100000001
01100000

Fig. 3. Mapping field-based regulation to a binary chromosome (i.e., design DNA) in a genetic algorithm.

To link the dDNA and the evaluation of the system's task completion, we use NetLogo software to model task completion and a separate program to control NetLogo and perform the GA operations. NetLogo (Wilensky, 1998*a*) is an agent-based modeling program. The NetLogo world consists of turtles (agents) moving over patches in the environment. The agents can be programmed with particular behaviors, and when many are placed into a single simulation, their emergent global behavior can be studied. NetLogo has a controlling application programming interface through which the agent simulation environment is linked to our GA Java program.

The GA program initially seeds a random population of genomes, which are a numeric encoding of the dDNA design parameters. Then, for each genome, it initializes a NetLogo simulation and sets the variables according to the dDNA. These variables represent the relative weights from the parametric behavioral model that agents use when calculating their local field values. NetLogo then runs the simulation for a specified number of time steps and reports the results to the GA. Finally, the GA interprets the global results according to a fitness function and applies the GA operators of selection, crossover, and mutation in order to create the next generation of genomes. This process is repeated until the final generation is reached.

5.1. COARM behavioral model

An in-depth description of flocking capabilities in a CSO system is given in (Chiang, 2012), where various relative weights given to flocking behaviors at the local level result in standard flocking, spreading, obstacle avoidance, or searching activity at the system level. This paper's flocking simulation was inspired by the work of Reynolds (1987) and Wilensky (1998*b*). The flocking mCells represent small robots moving on a two-dimensional surface in a torroidal (wrapped) world. Each mCell has a limited sensory range and very little memory. At each time step, an mCell will sense the positions and heading of all the other mCells within its radius of vision and react according to a set of predefined rules. These rules take the form of five competing desired step vectors. The five mCell behaviors are defined as

1. cohesion: step toward the center of mass of neighboring agents
2. avoidance: step away from agents that are too close
3. alignment: step in the direction that neighboring agents are headed
4. randomness: step in a random direction
5. momentum: step in the same direction as the last time step

The acronym COARM is used to refer to these behaviors (Chiang & Jin, 2011). An agent calculates the cohesion, avoidance, and alignment directions according to its locally sensed sField. Because this is a homogeneous system, the sField description includes only one type of relationship r_{11}, whose trigger is proximity (within three mCell diameters). Essentially, the mCell is responding to attraction (C) and nonlinear repulsion (O), and tries to match velocity (A) for each other cell in its neighborhood. The relevant state variables are the output of a random number generator for the randomness behavior, and an mCell's last step, which creates the momentum behavior. There are no tField objects:

$$b_{t+1} = \mathrm{FBR_{BS}}(\mathrm{FBR_{FT}}(\mathrm{FLD_S}(\mathrm{SNS_S}(a_{11}))), \mathrm{State}). \quad (9)$$

Figure 4 displays the spatial coordinates that an mCell's SNS operators obtain. The field transformation operator will assign a field value to every point within its stepsize limit. The field transformation functions are given in the following equations, based on sField and state, respectively:

$$\mathrm{FLD_s}(r, \theta, \phi) = -r + \frac{-1}{r} + \|v\|\cos(\theta - \phi), \quad (10)$$

$$\mathrm{FLD}(r, \theta, \phi) = \|v_0\|\cos(\phi) + \mathrm{step} \times \cos(\mathrm{RA} - \phi), \quad (11)$$

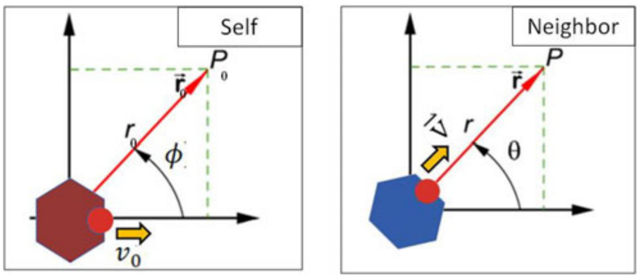

Fig. 4. A mechanical cell and neighbor frames of reference.

where v_0 is an mCell's current velocity (last step), RA is an angle output by a random number generator, step is the magnitude of the random behavior, ϕ is the angle from an mCell's current heading, v is a neighbor's velocity, θ is a neighbor's current heading relative to the mCell, and r is the scalar distance from the neighbor. Note that terms involving r are measured from the neighbor's position, and terms involving angles are measured with the mCell at the origin with a heading of 0 degrees.

The social field policies are applied to every neighbor in an mCell's radius of detection, and all terms from Equations (10) and (11) are added together in a parameterized sum to perform the $\mathrm{FBR_{FT}}$ operation:

$$\begin{aligned}\mathrm{FLD}(r, \theta, \phi) = {} & c \times \frac{-1}{N}\sum_{i\in\eta}^{N} r_i + o \times \frac{-1}{N}\sum_{i\in\eta}^{N}\frac{1}{r_i} \\ & + A \times \frac{1}{N}\sum_{i\in\eta}^{N}\|v_i\|\cos(\theta-\phi) + R \times \text{step} \\ & \times \cos(\mathrm{RA}-\phi) + M \times \|v_o\|\cos(\phi), \end{aligned} \tag{12}$$

where the COARM acronym returns, this time to represent the parameters in the field transformation equation. This equation maps each point in the plane to a preference value, creating the necessary (*behavior*, *preference*) set for behavior selection. The behavior selection is to simply move to the location within the maximum stepping distance with the highest preference. We allow the GA to change the relative weights of the COARM parameters, which will change the behavior of each mCell and consequently the emergent behavior of the system.

5.2. Flocking

To evolve flocking, an initial population of 15 candidate genomes was generated. The genomes consist of 40-bit binary strings, which can be parsed as five 8-bit, binary numbers. Each binary number sets the relative weight of a COARM parameter, and the parameters varied from 0.02 to 50. After mapping the genome to COARM parameter values, a simulation with these parameters was created in NetLogo. In the simulation, 30 mCells were initially placed on the field and allowed to run for 250 time steps.

The normalized total momentum of the system at the end of the run is used as the fitness function. This will have a value of 1.00 if all agents are moving in the same direction with maximum step size, and an expected value of 0.00 if agents are moving in random directions.

$$\vec{M} = \sum_{i=1}^{N}\vec{v}_i, \tag{13}$$

$$\text{fitness} = \frac{\|\vec{M}\|}{N}, \tag{14}$$

where N is the total number of mCells. Fitness scaling (Sadjadi, 2004) was used so that the best candidate in any generation had a fitness 30% greater than the average of all candidates in that generation. The best candidate at each generation was cloned to the next generation, and the remaining candidates were stochastically selected, with replacement, for single-point crossover with probability proportional to their fitness, until the next generation of 15 candidates was full. All nonclones were allowed to mutate, with a probability of 1% per bit of genome. This process was repeated for 40 generations. A full GA run required 11–14 min on a laptop computer with an Intel 2.2-GHz dual core processor and 4 GB of RAM.

Unrestricted, the GA will quickly maximize the relative weight of the alignment behavior for this fitness function. Figure 5 displays the evolution of the average parameter values among the 15 genomes at each generation. The relative weight of randomness was fixed at 1.00 because it was primarily the ratios of weights that mattered, not the absolute magnitude. The figure clearly shows a population takeover by systems with large alignment parameters.

Figure 6 displays the progression of fitness across generations for the same GA, where the fitness of the best candidate at each generation and the average of all candidates at each generation are plotted separately.

This optimization strategy seemed obvious, because the alignment behavior causes agents to match their heading with their neighbors. Given enough time, it seems intuitive that a system of agents focused on local alignment will eventually reach a state of global alignment, so in order to further challenge the algorithm, a new flocking behavioral model was established with the restriction that the relative weights of alignment and randomness must be equal (set to 1 in this instance). This makes the optimization much more difficult, because randomness counteracts the tendency toward coherent flocking that alignment builds. The results are given in Figures 7 and 8.

These results showed more inconsistency when compared to the unrestricted flocking example, but by the final generation, the best-of-generation candidates did display high fitness values, indicating that they were moving as a group with a single heading. The "strategy" found by the GA was to make both alignment and randomness mostly irrelevant by keeping the momentum weight very high. As long as the cohesion and avoidance balanced each other (it was sufficient that they differ by a factor of <5), the momentum could act as a system memory, allowing the alignment tendency to slowly build up during the course of the simulation, while the randomness effects cancelled themselves out.

5.3. Territory exploration

One of the goals in designing self-organizing systems is flexibility, so we attempted to use the same hardware assumptions and behavioral model as the flocking simulation to perform a different task: exploration. The change in functionality is a result of the change in relative parameter values. Here, 11 mCells are initially placed in a line in the center of the world and allowed to uncover white patches by moving near them

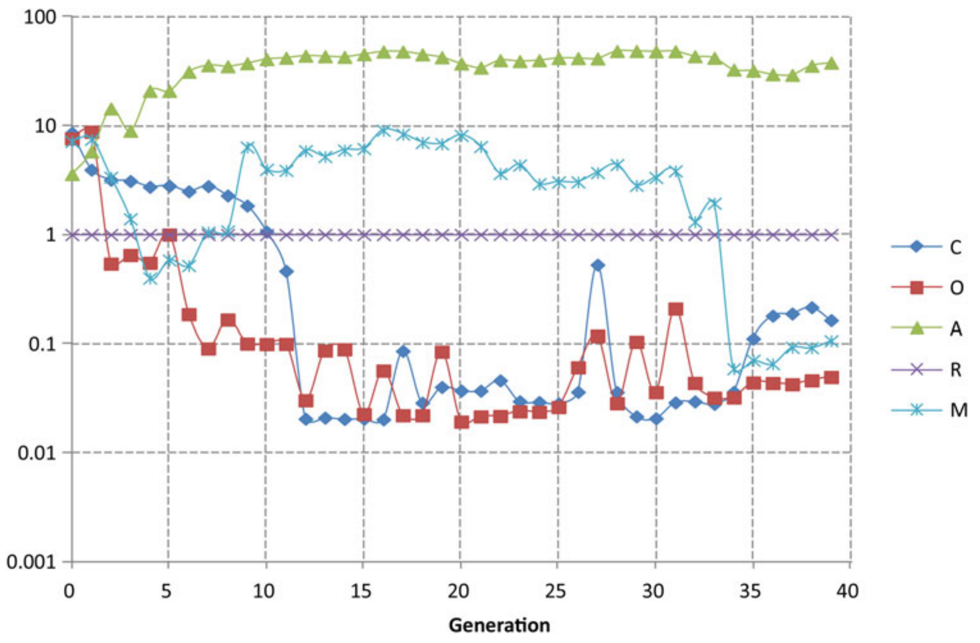

Fig. 5. Flocking average parameter values.

and darkening them. A run lasted 200 time steps, and the fitness of the run depended on how closely the system came to matching the desired amount of exploration. The desired exploration amounts ranged from 0% to 100%. Because GAs are stochastic, there is no guarantee that multiple runs will converge to the same set of parameters. While the results were mostly consistent in the flocking experiments, in the exploration experiments, qualitatively different behavioral strategies for the same FR were often evolved in different GA runs. These systems appear to work very differently, but their overall fitness scores are similar. Figures 9 through 12 show the behaviors evolved for 25% and 100% exploration, respectively.

It can be seen from these figures that two distinct successful behavioral profiles were developed for each of 25% and 100% exploration. The high-avoidance, high-momentum strategy for 100% exploration (Fig. 11) can be thought of as the intuitive strategy. The high avoidance and high momentum cause the mCells to immediately spread out in different directions and continue in straight lines until they come near each other, when they diverge again. This causes the

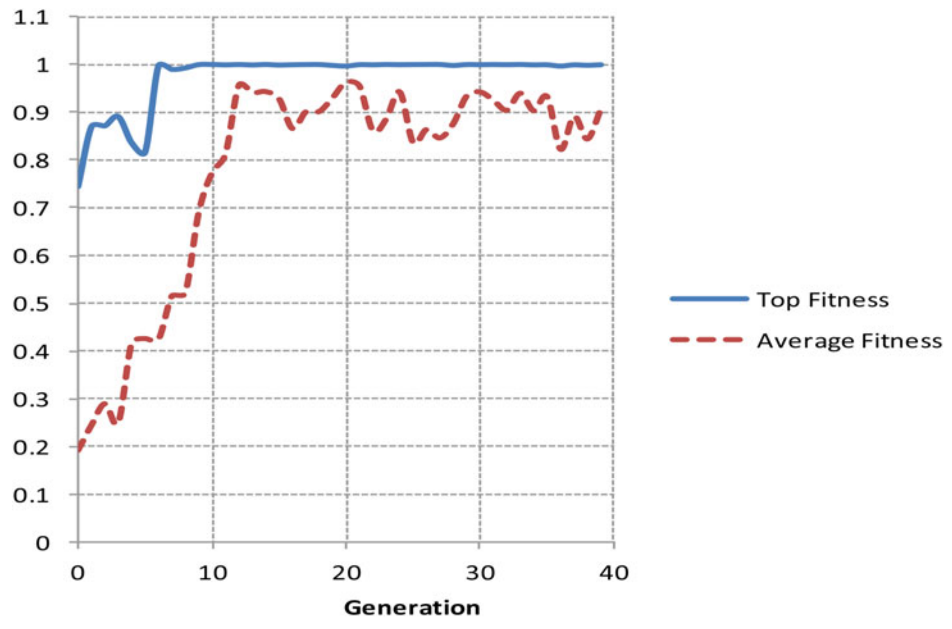

Fig. 6. Flocking fitness evolution.

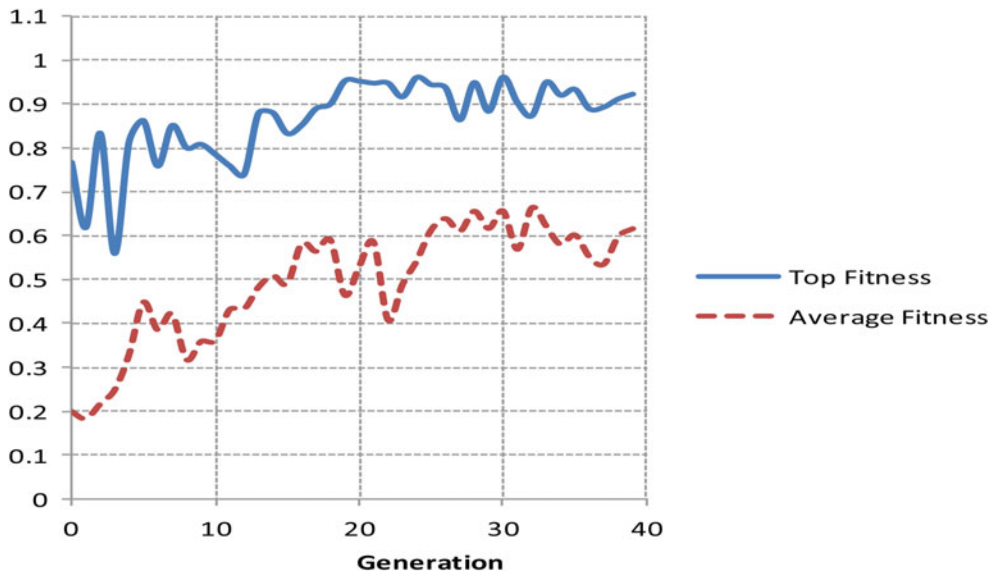

Fig. 7. Flocking fitness with equal alignment and randomness.

mCells to independently explore most of the field without wasting time by following one another and poring over trodden ground. The fanning and sweeping strategy of Figure 12 was actually more effective and relied on a novel combination of flocking and expansion. With high alignment, and a proper ratio between cohesion and avoidance, the mCells in this simulation were able to spread to the width of the arena and complete a full lap within the 200 time step limit, uncovering almost 95% of the field.

The 25% exploration task required a method of slowing down the exploration so that the system was not penalized for uncovering too much of the field (the mCells were forced to run for the full 200 time steps regardless of their progress). Figure 9 shows one successful strategy for halting exploration at the correct percentage. In this configuration, the mCells are assigned high avoidance and very low momentum. This causes an initial expansion, but the trajectories are unsustainable because there is no momentum to carry the mCells forward. Once they are out of one another's sensory radius, their randomness dominates because cohesion, avoidance, and alignment are only active when there are neighbors nearby. As a result, after the initial short expansion, the mCells tend to stay in place and randomly move about a point, not uncovering much new territory, so that by the end of the run, as a

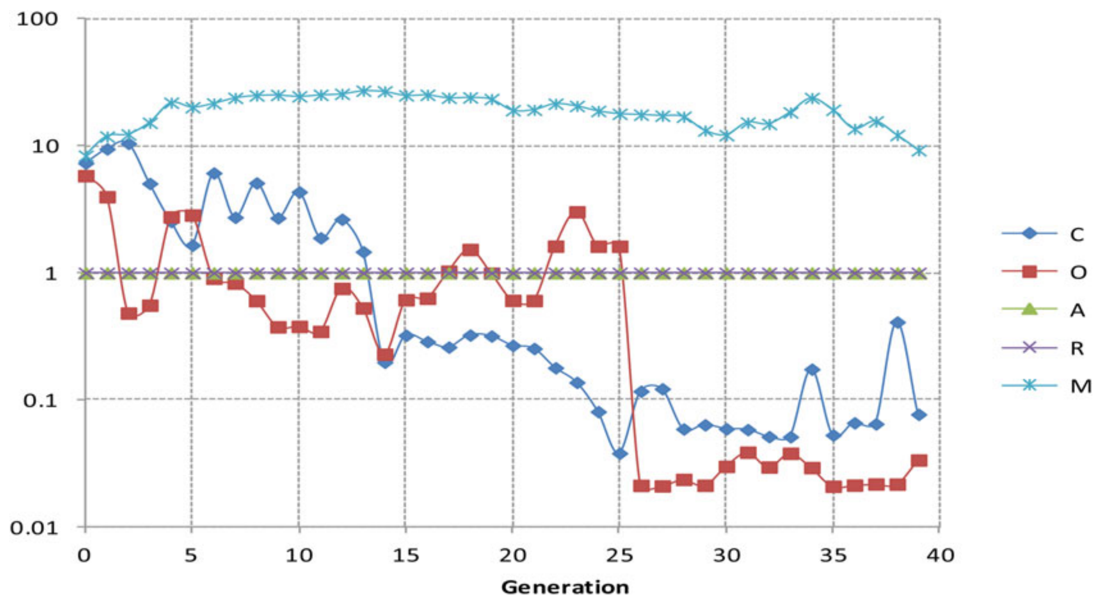

Fig. 8. Flocking parameter values with equal alignment and randomness.

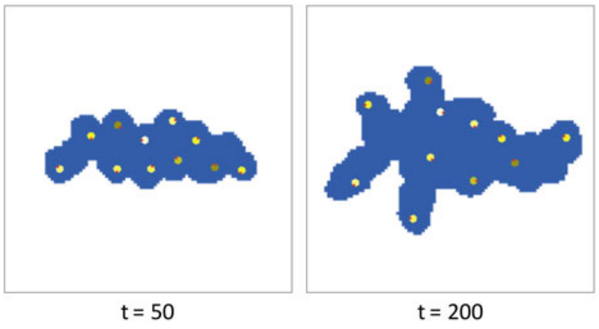

Fig. 9. High-avoidance, low-momentum for 25% exploration.

system they have uncovered approximately 25% of the field. Figure 10 shows a more novel behavioral strategy for 25% exploration that was evolved by some applications of the GA. In this case, the mCells' high alignment and cohesion tendencies caused them to form single-file lines. After the initial formation of the file, the swarm would extend out from its starting point and uncover a narrow swath of new territory. This system actually exploited the time limit as a resource, because it was still exploring new territory up until the cutoff, but the time required to initially form the structure and the slow rate of discovery reliably led it to discover close to 25% of the field just before the simulation stopped.

5.4. Foraging

Foraging is another task that can be performed by self-organizing systems. It is particularly interesting for us because, in addition to coordinating movements relative to other agents, agents in foraging must complete specific tasks of moving food from sources to home. With this case study, we intend to test how task requirements, such as moving food, can be modeled and captured by our task fields (tFields), and how such complex task fields can be coped with by more sophisticated relations (sFields) among the agents. Most foraging simulations are inspired by ants, which communicate via pheromones. Artificial systems could accomplish the foraging task using electromagnetic signaling. In our simulation, color signaling is assumed for the purpose of on-screen visualization. Our mCells are allowed to change color based on their current task and state. The parametric behavioral model is expanded so that mCells may have different reactions to other mCells of diverse states, a purposeful introduction of heterogeneity into the system.

The agents' visible states include location, color, and heading. The task object is the food, and the task goal is the home base where food is to be returned. The sField relationships are triggered by proximity within three mCell diameters. Food can be sensed within this same neighborhood, and the direction toward home can be sensed at all times. This simulates the situation where there is a central beacon (ants have the large concentration of pheromones emanating from their

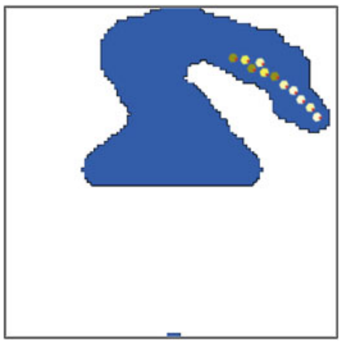

Fig. 10. High-alignment, high-cohesion for 25% exploration.

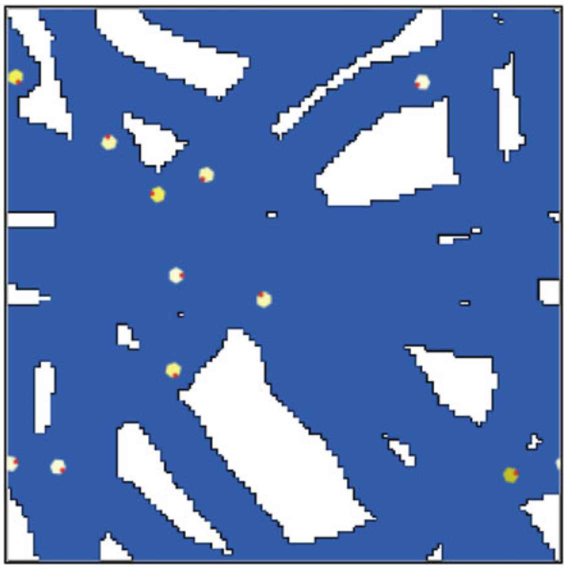

Fig. 11. High-avoidance for 100% exploration.

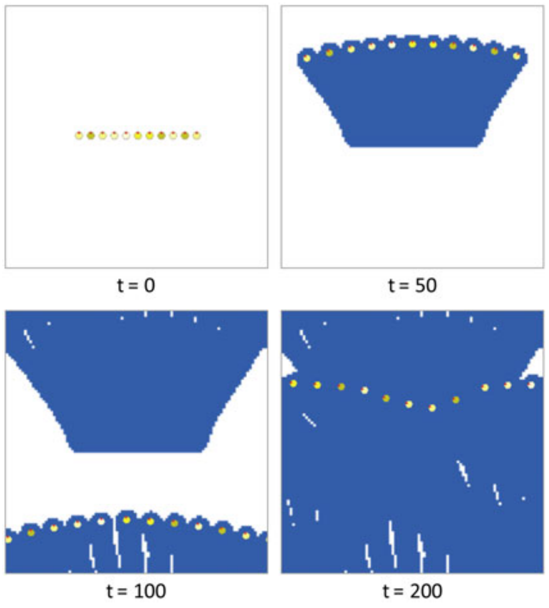

Fig. 12. High-alignment, high-momentum fan/sweep behavior for 100% exploration.

nest) signaling to all agents simultaneously, but not controlling any individual's actions. In a practical situation, the beacon could be broadcast from a boat at night in a search-and-rescue mission, from a disposal zone in a beach cleanup task, or from a silo in a harvesting system. The sField relationships are the same as in Equation (10), and the field transformations of tField and state parameters are modeled as:

$$\mathrm{FLD_T}(r, \theta, \phi) = \mathrm{step} \times (\cos(\mathrm{RA} - \phi) + \cos(f - \phi) + \cos(h - \phi)), \quad (15)$$

where f and h are the angles toward food (when sensed), and home, respectively.

Because an individual mCell's behavior must be different depending on whether it is carrying food, mCell behaviors are highly dependent on state. Because the mCells are allowed to differentiate, the description of the sField and the decision structure must be more complex than in the flocking and exploration tasks. Here, we assume that mCells will differentiate into two types: those with food and those without. Because we rely on the GA, it is feasible to define a large number of allowable relationships. We define the maximum number (4) based on the number of distinct agent types (2). In larger systems, it may be desirable to limit the interactions among different sets of agents. In any case, if the social topology is too interconnected for the task, we should expect the GA to minimize interaction magnitudes between agent types that should not be linked.

The policies defined for these relationships will be based on the cohesion, avoidance, and alignment parameters of the COARM model described earlier. Momentum is not used, and randomness is treated separately because it does not depend on agent interactions. Once an mCell has identified its neighbors and their type, it will apply the parameters of its policy for each neighbor. The mCells generate their tField by sensing the home beacon and food, and their reactions are simple attraction or repulsion. The tField and sField are added together by the mCells to form the behavioral profile:

$$\begin{aligned}\mathrm{FBR_{FT}}(r, \theta, \phi) = {} & C \times \frac{1}{N}\sum_{i\in\eta}^{N}(-r_i) + O \times \frac{1}{N}\sum_{i\in\eta}^{N}\frac{-1}{r_i} \\ & + A \times \frac{-1}{N}\sum_{i\in\eta}^{N}\|v_i\|\cos(\theta - \phi) \\ & + \mathrm{step} \times (R \times \cos(\mathrm{RA} - \phi) \\ & + F \times \cos(f - \phi) + H \times \cos(h - \phi)), \quad (16)\end{aligned}$$

where the relevant parameters are applied based on the mCell's state. If necessary, the summations are carried out twice with different parameters to account for a heterogeneous set of neighbors. Because the flocking behaviors are normalized before addition, a large flock with no food will have the same relative influence as a small flock with food and vice versa. Behavior selection is to simply choose the location within the maximum step size that has the highest field value.

When an mCell moves onto a patch that contains food, it extracts 5 units of food from the resource and changes its color to green. If it carries food back to the home base, it deposits the food and changes back to brown, and the simulation counts that toward the food-returned total. The fitness is then calculated according to

$$\mathrm{fitness} = \mathrm{food}_r + \frac{1}{N}\sum_{i=1}^{N}\mathrm{food}_{c_i}, \quad (17)$$

where the summation occurs over each mCell. Subscript r represents the food returned to home before the time limit, and subscript c represents the food being carried by the agents at the time limit. The summation in this equation is used to differentiate systems early in the GA, where only a few systems return any amount of food. The design gets "partial credit" for at least finding food, and this behavior is eventually combined with other positive behaviors to create more successful systems in later generations.

An initial population was randomly seeded as 50, 144-bit, binary strings. Every 8 bits of the genome corresponds to one of the 18 agent parameters of Table 1. With 18 parameters to optimize, an exhaustive search would be very computationally expensive. A naïve parameter sweep, with just three levels for each variable (e.g., low, medium, and high), would require more than 3.8 million simulation runs. The use of an optimization algorithm cuts this number down substantially while still generating capable candidates. All of these GA experiments used fewer than 10,000 repeated simulation runs. The binary numbers were mapped to decimal numbers between 0.02 and 50 for cohesion, avoidance, and randomness. Alignment and the tendency toward food and home varied between –50 and 50. The best candidate of each generation was cloned directly to the next generation, and the remaining candidates were created using the same fitness scaling, selection, and crossover of the previous case studies. The mutation probability was 0.5% per bit of genome for all nonclones. A

Table 1. *Foraging simulation parameters*

mCell State	Neighbor State	Cohesion	Avoidance	Alignment	Randomness	Home	Food
Food	Food	C_1	O_1	A_1	R_1	H_1	F_1
	No food	C_2	O_2	A_2			
No food	Food	C_3	O_3	A_3	R_2	H_2	F_2
	No food	C_4	O_4	A_4			

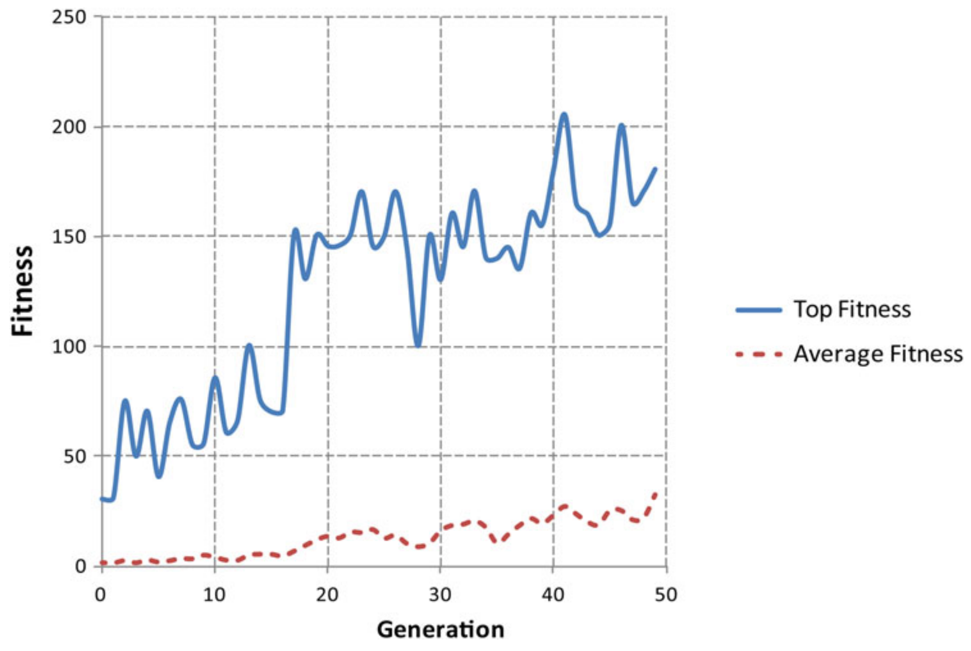

Fig. 13. Foraging experiment fitness evolution.

100-candidate, 50-generation (5000 total fitness evaluations) GA run required approximately 14 h to complete.

It can be seen from Figure 13 that the GA showed continual improvement in the best-of-generation fitness and average fitness values. The best candidate of the first generation returned 30 units of food (6 round trips), and the best of the final generation returned 180 units of food (36 round trips). The best fitness found in any generation was 205 in the 41st generation. Owing to the elitism component of the GA, this candidate was cloned to the next generation, but it was unable to reliably reproduce such great results. It was eventually replaced, although its genes did propagate to future candidates. Other GA runs produced qualitatively similar results, with no obvious improvement in runs lasting longer than 50 generations. Because there were 18 variables to optimize, and the populations showed considerable diversity, it is not illuminating to show a plot of the average parameter values across generations. Instead, we look at the randomly chosen parameters of the best candidate of the first generation and compare them to the evolved parameters of the last generation's top candidate.

Table 2 shows the parameter values of the first generation's most successful candidate. The simulation screenshots in Figure 14 illustrate the behavior of this system. This candidate's behavior relied on a very high tendency toward home when an agent had food (H_1), and a very high tendency away from home when an mCell did not have food (H_2). This was enough to place one or two mCells on a straight line between the base and the food, allowing them to make about 6 round trips. The high cohesion/avoidance ratio among the mCells without food led to the ineffective grouping at the right edge of the field.

Table 3 shows the parameters that were evolved for the best candidate in the final generation. Note the very large negative alignment parameter between agents that both had no food (A_4). Generally, the mCells with food have a strong tendency to align with one another, while the mCells without food have a strong tendency to maintain opposite headings. The magnitudes of the interactions between agents of different states are small.

The "strategy" evolved was to use the edges of the arena to guide the agents toward the food, because the food was placed in a corner opposite the home base. To accomplish this, the mCells without food had negative alignment values toward one another. This caused an initial shuffling period because the group could not reach an equilibrium flocking heading. Eventually the mCells spread far enough apart that their

Table 2. *Foraging parameters for best candidate of first generation*

mCell State	Neighbor State	Cohesion	Avoidance	Alignment	Randomness	Home	Food
Food	Food	0.189	0.208	20	0.492	30.7	−7.07
	No food	27.8	4.96	28.8			
No food	Food	0.0359	0.131	−0.737	5.28	−25.5	−1.44
	No food	2.23	0.0707	−2.08			

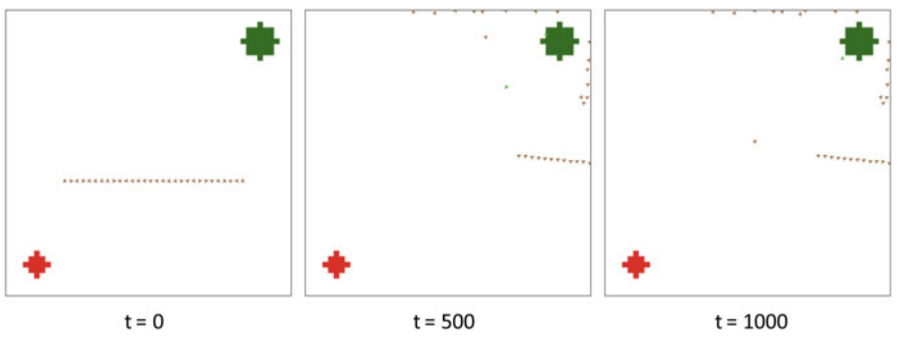

Fig. 14. Foraging behavior for best candidate of first generation. Home is at bottom left, and the food is at top right.

negative home tendency dominated and drove them toward the top and right edges of the field, with a few mCells randomly finding the food area. The system eventually reached a state where most mCells without food were on an edge, but the negative alignment values ensured that they did not get stuck. If a new mCell arrived at the edge near another, one would have to change direction so that they could maintain opposite headings. This caused a chain reaction of mCells bumping each other off the edge until one reached the food, when its strong positive home tendency would take over. After returning the food, an mCell's negative home tendency would cause it to move toward an edge again, starting another chain reaction. This configuration persisted until the 1000 step time limit, as shown in Figure 15, allowing the system to return 180 units of food. In these lines of mCells on the edge, we see the social field giving rise to task-based structure that allowed the system to complete its FR.

5.5. Discussion

5.5.1. *Effect of two-field based approach*

We have shown that the interaction of two fields can lead to emergent task completion in self-organizing systems. In the flocking and exploration tasks, mCells relied entirely on sField relations, and for the foraging task, both sField and tField were important. Through evolutionary synthesis, the GA was able to develop dDNA that corresponded to the task environment, FR, and agent relations. In the exploration task, the GA implicitly embedded knowledge about the size of the field to be explored by selecting candidate systems whose social relations would cause them to assume proper shapes and sizes in order to uncover the correct amount of territory. The foraging task saw the evolution of a strategy that integrated task and social fields so that the mCells not only formed the proper structure but also formed it in the proper place. In this simulation, the agents formed lines along the boundary of the arena to funnel one another toward the food source. Here we see the tField creating an attractor at the boundary of the arena, while the sField turns this aggregation of mCells into a path toward the goal.

5.5.2. *Generational learning*

These strategies are the result of generational learning caused by the repeated trial-and-error of the GA. The GA was shown to embed global knowledge into the mCell dDNA. For example, in the exploration task (Fig. 12), the global value of the field width was embedded in a cohesion/avoidance ratio that allowed the mCells to spread to that width or in a proper momentum weight, which caused motion to die out after enough spreading. In the foraging task, the location of the food relative to home was embedded in the mCells' home aversion (go right/up) and antialignment (go up/right). When this information is embedded as dDNA, it is not necessary that agents have any specific global knowledge, only that they can recreate the proper structures required through cooperative, iterative application of local rules.

5.5.3. *Validation and repeatability*

These local rules were successfully tuned by a GA, but an engineer must take a critical look at the results from any automated design algorithm, because computational synthesis cannot yet replace the human designer entirely. Some results may be spurious or optimized to the peculiarities of the fitness

Table 3. *Foraging parameters for best candidate of final generation*

mCell State	Neighbor State	Cohesion	Avoidance	Alignment	Randomness	Home	Food
Food	Food	0.102	0.94	21.2	0.0393	5.21	1.733
	No food	0.0241	0.422	−0.376			
No food	Food	0.148	1.2	−0.783	5.44	−5.21	9.59
	No food	13.29	33.49	−47			

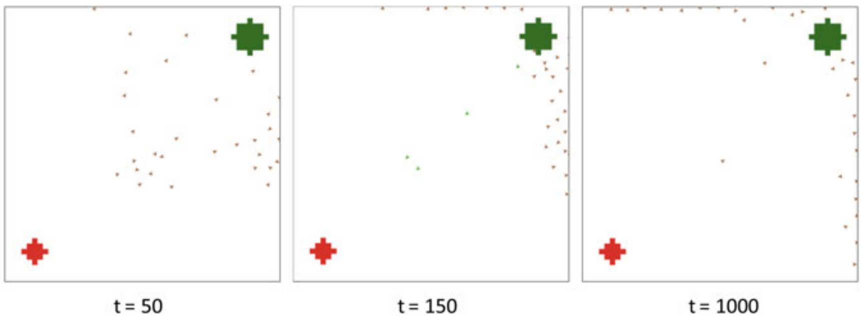

Fig. 15. Foraging behavior for best candidate of final generation.

function rather than the real-world task requirements. Designers must be careful to ensure that the GA is truly optimizing for the attributes that the system will need in the real world because there is a danger that the GA will simply take advantage of quirks in the initial condition or fitness function. For example, the 200 time step limit in the exploration experiment had a substantial effect on the evolved behavior. This limit was at times a help or a hindrance. The limit was an obstacle for the high-avoidance, high-momentum 100% exploration strategy, because this system would only reliably uncover about 85% of the field before it was stopped. It had little effect on the high-avoidance, low-momentum 25% exploration strategy, because these agents achieved 25% exploration and then largely ceased to uncover new territory well before the time limit hit. The time limit was actually exploited as a resource by the high-cohesion, high-alignment 25% exploration strategy, because these agents were continuously uncovering new territory, but the time limit cut them off close to the time they discovered the 25th percent of the field, keeping them from discovering too much. All of the strategies evolved performed well with an exact 200 time step limit, but the high-cohesion, high-alignment 25% strategy would have performed poorly if the limit were any greater or smaller. The designer in this case must clarify whether the 200 time step limit is a hard constraint or simply an estimation of the real-world scenario.

Similarly, for 100% exploration, the fanning and sweeping strategy was dependent on the width of the field and the initial formation of the agents. If this initial formation can be replicated in actual deployment, this strategy is probably the best, but the more random strategy may be necessary if the initial formation is not so regular. For this paper, we simply remind the designer to consider these possible optimization pitfalls and leave the determination of real versus spurious results to the designer's discretion, but as future work, we plan to build GAs that can capture and save qualitatively distinct behavioral strategies for later analysis and to eliminate possible loopholes for the GA to exploit through more realistic world building.

In this paper, where only one set of results was presented for a case study, repeated GA runs led to qualitatively repeatable results, with the same pattern of ratios among the various parameters being found by each GA, and the emergent behavior appearing quite similar. The exact numbers would vary slightly, because the GA is partially stochastic, and there is some inherent stochasticity in the behavioral models and simulation environment. This is a trade-off that must be made when using any stochastic optimization method. The GA's results can also be affected by its own parameters, such as fitness scaling and crossover percentage, which were not varied in our study. Occasionally, meta-optimizations are used to set these values, but for our work, we set them based on an informal pilot study.

Repeatability of any particular set of optimized agent parameters was not found to be an issue. This is because clones and near clones of the best candidates are continually being retested across generations. Thus, only the reliably successful candidates survive to the final generation. This is a useful result from the use of evolutionary optimization, but it should not be generalized or taken for granted. A designer of self-organizing systems will always need to carefully evaluate the repeatability of system behavior, whether empirically or through mathematical proofs of convergence.

5.5.4. GA as design guide

At its worst, a GA will evolve a trivial set of parameters to "deceive" the fitness function and offer no practical use, but at its best, a GA can allow a designer to quickly search a design space and even highlight errors in the original problem specification. In the foraging task, all the successful candidate solutions evolved negative alignment behaviors between mCells that did not have food. This meant that the mCells were actively trying to set their heading to the opposite of their neighbors, preventing any flocking. It could have been useful for the mCells to flock together, but the negative alignment was used to correct for a design error: the lack of boundary detection.

The mCells were not endowed with the ability to detect or react to the boundary of the arena, so many would simply move to an edge and stay there without doing any useful work. The negative alignment helped to rescue some mCells stuck on the edge, because if a new mCell reached the edge close to another, one would immediately turn away from the boundary so they could maintain opposite headings. This gave an mCell a chance to move back toward the center of the field. Some systems showed a chain reaction of this behavior, where this disturbance would slowly move along a boundary until an mCell found food and changed its state.

Because the food was placed in the corner of the arena, the most successful systems actually used this behavior to send mCells along the edges to the food source, a strategy unanticipated by the human designer of the simulation. Because the GA evolved behavior that consumed so many resources to overcome the boundary problem, a designer could use that as evidence that basic boundary detection should be embedded into the mCells.

6. CONCLUSIONS AND FUTURE WORK

The CSO system has been proposed as a bottom-up approach to building complex systems. Developing a general behavioral model of agents and a synthesis method has been a major challenge in CSO research. In this paper, we propose a two-field based model to characterize agents' behavior. The task field captures the task environment while the social field arises from agent–agent relations. A GA-based computational synthesis method is presented that evolves effective self-organizing mechanisms and behavioral rules for agents based on the task and social fields. The case studies have demonstrated the effectiveness of the proposed model and the GA-based computational synthesis approach.

This research affects the field of engineering design in several ways. The case studies provide several examples of integrated simulation and optimization, which can be used in the design of many systems. The field-based behavioral model presented here can be applied to existing distributed systems for simulation and analysis. For example, as passenger vehicles gain the ability to locally communicate information about their velocity and location (Rogers, 2014), a city's emergent traffic patterns could be modeled with high fidelity using agents with field-based behavior. The framework is meant to parallel traditional design from a biological perspective, by using a cell-based, bottom-up approach, rather than a component-based, top-down approach. The design and deployment of large-scale self-organizing systems is still a long-term goal, and this research is not meant to compete with traditional, top-down design in the short term or for simple products. It is meant to exist alongside conventional design to aid in the design of distributed and adaptable systems, or for the modeling and analysis of existing complex systems.

Our ongoing and future work aims to address three issues. First, there are many possible GA upgrades to produce superior results (e.g., different methods of selection, mutation, and crossover; Goldberg, 2002). In addition, it is helpful for the GA to maintain diversity within the pool of candidates so that qualitatively different behavioral strategies that give the same results are preserved, rather than having a single strategy in a homogeneous population at the end of a GA run. More advanced fitness functions will be developed that consider design factors beyond performance, such as cost and manufacturability.

Second, there is a potentially rich space to explore for parametric behavior models (PBMs). The PBM should allow for much more than these specific flocking-based studies. Using only the COARM model restricted our search space of all possible behaviors. The PBM rests on a parametric description of field formation and agent–agent relationships. Cast in this light, the conceptual design of CSO systems may be thought of as a search through the PBM space for a minimal set of mCell behaviors and relations. The GA then tunes the parameters within this PBM. A regimented analysis of this model may help designers of self-organizing systems to make more deliberate decisions when performing global-to-local mapping, rather than making intuitive guesses.

Third, the case studies show the flexibility of CSO systems but do not explore robustness and resilience in depth. Further case studies could be tailored to test for these capabilities by rearranging the arena and randomly deactivating agents. Adding these adversities in a way that is random but "fair" to the mCells is challenging.

Building a physical swarm of self-organizing robots to test the principles on real hardware, rather than just in simulation, has been our ongoing work, and there is progress being made along this line in our laboratory.

ACKNOWLEDGMENTS

We thank the guest editors and anonymous peer reviewers for their helpful and insightful comments. This paper is based on the work supported in part by the National Science Foundation under Grants CMMI-0943997 and 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

Ashby, W.R. (1958). Requisite variety and its implications for the control of complex systems. *Cybernetica*, *1(2)* 83–99.

Bai, L., & Bree, D. (2012). Chemotaxis-inspired cellular primitives for self-organizing shape formation. In *Morphogenetic Engineering: Toward Programmable Complex Systems* (Doursat, R., Sayama, H., & Michel, O., Eds.), pp. 141–156. Berlin: Springer–Verlag.

Beckers, R., Holl, O.E., Deneubourg, J.L., Bielefeld, Z., & Bielefeld, D. (1994). *From Local Actions to Global Tasks: Stigmergy and Collective Robotics*, pp. 181–189. Cambridge, MA: MIT Press.

Beni, G. (1988). The concept of cellular robotic system. *Proc. IEEE Int. Symp. Intelligent Control, 1988*, pp. 57–62.

Bentley, P. (1999). Three ways to grow designs: a comparison of embryogenies for an evolutionary design problem. *Proc. Genetic and Evolutionary Computation Conf.*, pp. 35–43. San Francisco, CA: Morgan Kaufmann.

Bentley, P. (2001). *Digital Biology: How Nature Is Transforming Our Technology and Our Lives*. New York: Simon & Schuster.

Calvez, B., & Hutzler, G. (2006). Automatic tuning of agent-based models using genetic algorithms. In *Multi-Agent-Based Simulation VI* (Sichman, J., & Antunes, L., Eds.), Vol. 3891, pp. 41–57. Berlin: Springer.

Chen, C., & Jin, Y. (2011). A behavior based approach to cellular self-organizing systems design. *Proc. ASME 2011 Int. Design Engineering Technical Conf. & Computers and Information in Engineering Conf.*, Washington, DC.

Chiang, W. (2012). *A meta-interaction model for designing cellular self-organizing systems*. PhD Thesis. University of Southern California, Los Angeles.

Chiang, W., & Jin, Y. (2011). Toward a meta-model of behavioral interaction for designing complex adaptive systems. *Proc. ASME 2011 Int. Design Engineering Technical Conf. & Computers and Information in Engineering Conf.*, pp. 1077–1088, Washington, DC.

Crutchfield, J.P., Mitchell, M., & Das, R. (1996). Evolving cellular automata with genetic algorithms: a review of recent work. *Proc. 1st Int. Conf. Evolutionary Computation and Its Applications*, Moscow.

Cucker, F., & Smale, S. (2007). Emergent behavior in flocks. *IEEE Transactions on Automatic Control 52(5)*, 852–862.

Doursat, R. (2011). The myriads of Alife: importing complex systems and self-organization into engineering. *Proc. 2011 IEEE Symposium on Artificial Life, ALIFE*, pp. 1–8.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed. Reading, MA: Addison–Wesley Professional.

Goldberg, D. (2002). *The Design of Innovation*, 1st ed. Berlin: Springer.

Goldstein, S., & Mowry, T. (2004). Claytronics: a scalable basis for future robots. *Proc. Robosphere*. Mountain View, CA: NASA Ames Research Center.

Haken, H. (1978). *Synergetics: An Introduction: Nonequilibrium Phase Transitions and Self-Organization in Physics, Chemistry, and Biology*. Berlin: Springer–Verlag.

Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA: MIT Press.

Humann, J., & Jin, Y. (2013). Evolutionary design of cellular self-organizing systems. *Proc. ASME 2013 Int. Design Engineering Technical Conf. Computers and Information in Engineering Conf.*, Portland, OR.

Jin, Y., & Chen, C. (2012). Field based behavior regulation for self-organization in cellular systems. *Proc. Design Computing and Cognition Conf. DCC'12*.

Jin, Y., & Li, W. (2007). Design concept generation: a hierarchical coevolutionary approach. *Journal of Mechanical Design 129(10)*, 1012.

Kelly, K. (1994). *Out of Control: The New Biology of Machines, Social Systems and the Economic World*. Reading, MA: Addison–Wesley.

Korb, J. (2011). Termite mound architecture, from function to construction. In *Biology of Termites: A Modern Synthesis* (Bigness, D.E., Roisin, Y., & Lo, N., Eds.), pp. 349–373. Dordrecht: Springer.

Mitchell, M., Crutchfield, P., & Hraber, P. (1994). Evolving cellular automata to perform computations: mechanisms and impediments. *Physica D: Nonlinear Phenomena 75(1–3)*, 361–391.

Payton, D., Daily, M., Estowski, R., Howard, M., & Lee, C. (2001). Pheromone robotics. *Proc. SPIE 4195, Mobile Robots XV and Telemanipulator and Telepresence Technologies VII,* p. 67.

Reynolds, C.W. (1987). Flocks, herds, and schools: a distributed behavioral model. *ACM SIGGRAPH Conf. Proc.*, pp. 25–34.

Rogers, C. (2014, February 3). "U.S. to propose vehicle-to-vehicle, crash-avoidance systems." *Wall Street Journal*. Accessed at http://online.wsj.com/news/articles/SB10001424052702303942404579360972335289080

Rubenstein, M., Ahler, C., & Nagpal, R. (2012). Kilobot: a low cost scalable robot system for collective behaviors. *Proc. 2012 IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 3293–3298.

Sadjadi, F. (2004). Comparison of fitness scaling functions in genetic algorithms with applications to optical processing. In *Optical Science and Technology: The SPIE 49th Annual Meeting*, pp. 356–364. Denver, CO: International Society for Optics and Photonics.

Shen, W.-M., Salemi, B., & Will, P. (2002). Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots. *IEEE Transactions on Robotics and Automation 18(5)*, 700–712.

Song, Y., Kim, J.-H., & Shell, D. (2012). Self-organized clustering of square objects by multiple robots. In *Swarm Intelligence* (Dorigo, M., Birattari, M., Blum, C., Christensen, A., Engelbrecht, A., Groß, R., & Stützle, T., Eds.), Vol. 7461, pp. 308–315. Berlin: Springer.

Sridharan, P., & Campbell, M.I. (2005). A study on the grammatical construction of function structures. *Artificial Intelligence for Engineering, Design Analysis and Manufacturing 19(3)*, 139–160.

Stonedahl, F., & Wilensky, U. (2010). Finding forms of flocking: evolutionary search in ABM parameter-spaces. *Proc. MABS Workshop, 9th Int. Conf. Autonomous Agents and Multi-Agent Systems*.

Thompson, J. (1967). *Organizations in Action*. New York: McGraw–Hill.

Trianni, V. (2008). *Evolutionary Swarm Robotics: Evolving Self-Organising Behaviours in Groups of Autonomous Robots*. Berlin: Springer.

Ueyama, T., Fukuda, T., & Arai, F. (1992). Structure configuration using genetic algorithm for cellular robotic system. *Proc. IEEE/RSJ Int. Conf. Intelligent Systems*, Vol. 3, p. 1542.

Van Berkel, S., Turi, D., Pruteanu, A., & Dulman, S. (2012). Automatic discovery of algorithms for multi-agent systems. *Proc. 14th Int. Conf. Genetic and Evolutionary Computation Conf. Companion*, pp. 337–344. New York: ACM.

Werfel, J. (2012). Collective construction with robot swarms. In *Morphogenetic Engineering: Toward Programmable Complex Systems* (Doursat, R., Sayama, H., & Michel, O., Eds.), pp. 115–140. Berlin: Springer–Verlag.

Werfel, J., & Nagpal, R. (2006). Extended stigmergy in collective construction. *Intelligent Systems, IEEE 21(2)*, 20–28.

Wilensky, U. (1998*a*). *NetLogo*. Evanston, IL: Northwestern University, Center for Connected Learning and Computer-Based Modeling. Accessed at http://ccl.northwestern.edu/netlogo

Wilensky, U. (1998*b*). *NetLogo Flocking Model*. Evanston, IL: Northwestern University, Center for Connected Learning and Computer-Based Modeling. Accessed at http://ccl.northwestern.edu/netlogo/models/Flocking

Yogev, O., Shapiro, A.A., & Antonsson, E.K. (2008). Engineering by fundamental elements of evolution. *Proc. ASME 2008 Int. Design Engineering Technical Conf., IDETC/CIE*.

Zouein, G., Chen, C., & Jin, Y. (2010). Create adaptive systems through "DNA" guided cellular formation. *Proc. Design Creativity 2010*.

James Humann is a doctoral candidate at the University of Southern California. He received a bachelor of science in mechanical engineering from the University of Oklahoma in 2010 and a master of science in mechanical engineering design from the University of Southern California in 2012. His research is focused on mechanical design, specifically the design of self-organizing systems, and related fields such as agent-based modeling, optimization, and complex systems engineering.

Newsha Khani is a doctoral candidate at the University of Southern California. She received her master of science degree in 2009 from Oregon State University and her bachelor of science in 2007 from the University of Tehran. Her research interests include design theory and methodology, and coordination and self-organization in multiagent systems.

Yan Jin is a Professor of aerospace and mechanical engineering at University of Southern California. He received his PhD in naval architecture and ocean engineering from the University of Tokyo. Prior to joining the University of Southern California faculty in 1996, he worked as a Research Scientist at Stanford University for 5 years. Dr. Jin is a recipient of the NSF CAREER Award (1998), the TRW Excellence in Teaching Award (2001), and the Xerox Best Paper Award (ASME Design Theory and Methodology Conference, 2002). He currently serves as Editor in Chief of *AIEDAM* and as a member of the editorial board of *Advanced Engineering Informatics* and *International Journal of Design Creativity and Innovation*. His current research focuses on self-organizing systems and design creativity.