



# Impact of Task Constraint on Agent Team Size of Self-Organizing Systems Measured by Effective Entropy

Hao Ji

Center for Advanced Research Computing,  
University of Southern California,  
3434 South Grand Avenue, Building CAL,  
Los Angeles, CA 90089  
e-mail: haoji@usc.edu

Yan Jin<sup>1</sup>

Department of Aerospace and Mechanical  
Engineering,  
University of Southern California,  
3650 McClintock Avenue, OHE 430,  
Los Angeles, CA 90089  
e-mail: yjin@usc.edu

*Self-organizing systems can perform complex tasks in unpredictable situations with adaptability. Previous work has introduced a multiagent reinforcement learning-based model as a design approach to solving the rule generation problem with complex tasks. A deep multiagent reinforcement learning algorithm was devised to train self-organizing agents for knowledge acquisition of the task field and social rules. The results showed that there is an optimal number of agents that achieve good learning stability and system performance. However, finding such a number is nontrivial due to the dynamic task constraints and unavailability of agent knowledge before training. Although extensive training can eventually reveal the optimal number, it requires training simulations of all agent numbers under consideration, which can be computationally expensive and time consuming. Thus, there remains the issue of how to predict such an optimal team size for self-organizing systems with minimal training experiments. In this article, we proposed a measurement of the complexity of the self-organizing system called effective entropy, which considers the task constraints. A systematic approach, including several key concepts and steps, is proposed to calculate the effective entropy for given task environments, which is then illustrated and tested in a box-pushing case study. The results show that our proposed method and complexity measurement can accurately predict the optimal number of agents in self-organizing systems, and training simulations can be reduced by a factor of 10.*  
[DOI: 10.1115/1.4065343]

**Keywords:** complex systems, self-organizing system, entropy, multiagent deep reinforcement learning, artificial intelligence, engineering informatics, machine learning for engineering applications

## 1 Introduction

Self-organizing systems (SOS) can consist of simple agents that work cooperatively to achieve complex system-level behaviors without requiring global guidance. The design of SOS takes a bottom-up approach, and the top-level system complexity can be accomplished through local agent interactions [1,2]. Complex system design by applying an SOS approach has many advantages, such as scalability, adaptability, and reliability [3,4]. Moreover, compared to traditional engineering systems with centralized controllers, SOS can be more robust to external changes and more resilient to system damage or component malfunctions [5–7].

Recent advances in deep reinforcement learning (RL) have made it possible to shift the focus of SOS design from field based and rule based to learning based. Making agents learn reduces human trial-and-error-based design effort as RL agents can explore and learn by themselves during training. After training, the learned

knowledge is stored in neural networks and utilized during application. For example, the independent multiagent RL approach has been taken to capture the self-organizing knowledge for agent behavior regulation in SOS design [8,9]. During the multiagent RL, each agent can be trained using its own independent neural network. This approach solves the problem of the curse of dimensionality of the action space when applying single-agent RL to multiagent settings. The results from our previous work indicated that there is an optimal number of agents that achieve the best learning performance [8,9]. However, finding such a magical number is computationally expensive as tens of thousands of training and testing simulations must be conducted for all agent numbers. There remains an issue of how to find the optimal agent number within a minimal number of training iterations.

In this study, we addressed the following research questions:

- Q1: What are the main factors that determine the optimal team size of an agent-based self-organizing system?
- Q2: How can we identify and analyze these factors to predict the best agent team size?
- Q3: To what extent can the prediction process be made computationally reasonable?

<sup>1</sup>Corresponding author.

Manuscript received February 13, 2023; final manuscript received March 23, 2024; published online May 31, 2024. Assoc. Editor: Yan Wang.

In the rest of this article, Sec. 2 reviews the relevant work in artificial self-organizing systems, multiagent RL, and complexity theory. In Sec. 3, task constraints are discussed, and a measure of effective entropy is presented with relevant concepts and steps for optimal team size prediction. Section 4 presents the computational method and a box-pushing case study using the proposed measurement for optimal team size prediction. The findings and insights of this work are discussed in Sec. 5, and the conclusions are presented in Sec. 6, together with future work directions.

## 2 Related Works

**2.1 Artificial Self-Organizing Systems and Multiagent Reinforcement Learning.** An artificial self-organizing system is designed by humans and has emergent behavior and adaptability, similar to nature [1]. Much research has been conducted on the design of artificial self-organizing systems. Khani et al. developed a social rule-based regulation approach in enforcing the agents to self-organize and push a box toward the target area [5,6]. Swarms of UAVs can self-organize based on a set of cooperation rules and accomplish tasks such as target detection, collaborative patrolling, and shape formation [10–13]. The robotic implementations mentioned earlier demonstrate the potential for building self-organizing systems. Still, the design methods of self-organizing systems [5,6,14] have drawbacks: designers must possess extensive domain knowledge to develop models or draw inspiration from nature, which is time consuming and hardly extendable to different scenarios. Also, generating design rules from global requirements to local interactions remains challenging.

Recent advances in multiagent RL have made it possible to shift the focus of self-organizing system design from rule based to learning based. Multiagent RL applies to multiagent settings and is based mainly on single-agent RL, such as Q-learning, policy gradient, and actor-critic [15,16]. Compared to single-agent RL, multiagent learning is faced with a nonstationary learning environment due to the simultaneous learning of multiple agents.

Multiagent systems can be classified into cooperative, competitive, mixed cooperative, and competitive categories [17]. SOS design focuses on cooperative agents because they share the same task goals. One natural approach for multiagent RL is optimizing each individual policy or value function. The most commonly used value function-based multiagent learning is independent Q-learning [18]. It trains each individual state-action value using Q-learning [18,19] and serves as a common benchmark in the literature. Tampuu et al. [17] extended the previous Q-learning to deep neural networks and applied a deep Q-learning network (DQN) [20] to train two independent agents playing the game Pong [17]. Foerster et al. applied the COMA framework to train multiple agents to play StarCraft games with centralized critics to evaluate decentralized actors [21]. In another study by Foerster et al., he analyzed his replay stabilization methods for independent Q-learning based on StarCraft combat scenarios [22].

As a multiagent environment is usually partially observable, Hausknecht and Stone [23] used deep recurrent networks such as LSTM (long short-term memory) [24] or GRU (gated recurrent unit) [25] to speed up learning when agents are learning over long periods. Lowe et al. developed multiagent deep deterministic policy gradient and tested their algorithm in predator-prey, cooperative navigation, and other environments [26]. Brown and Sandholm developed a superhuman AI model called “Pluribus” and showed that it could defeat top human professionals in six-player-no-limit Texas hold ‘em poker [27]. Baker et al. demonstrated that multiple agents could create a self-supervised auto-curriculum, generating various rounds of emergent strategy in hide-and-seek games [28]. Wu et al. developed Bayesian delegation, a decentralized algorithm that makes inferences like human observers about the intent of others. They successfully tested their algorithm in a cooking video game, “Overcooked” [29]. Despite the development of numerous multiagent RL algorithms, training multiple agents with such algorithms takes extensive time, which

makes finding the optimal team size computationally expensive. Predicting the optimal team size of SOS with minimal training experiments remains challenging.

**2.2 Complexity Measurements.** A complex system is a system that consists of many interacting components and whose collective behavior is challenging to model due to interdependencies or interactions between its parts [30]. Researchers in the mechanical engineering field have long addressed the issue of designing complex systems [31–36]. A self-organizing system is an example of such a complex system, with complex interactions between its agents.

Since Simon defined complexity as a property of the system that arises from the interactions of its parts in nonsimple ways [37], researchers from different fields have sought to measure system complexity [38–40]. More than 50 complexity measures have been proposed in the literature on engineering design and system engineering literature [39,41–43]. While there is no agreed-upon methodology for measuring complexity and what a good complexity measure should entail, there are some commonly held beliefs regarding which system attributes drive complexity. For example, there is broad agreement among the research community that the complexity of an engineered system is driven by three main system properties: size, interconnection, and structure [44]. System complexity grows as the system size increases because larger systems can generate more unique system states [37,45–47]. It is also agreed that increasing the interconnections within a system increases its complexity [37,45–47]. Structuring or minimizing random interconnections between system components, such as introducing a modular structure, reduces the system’s complexity [48–51].

Considerable research has been conducted on measuring complexity. Min et al. analyzed the structural complexity of complex engineering systems at various decomposition levels [52]. McCabe’s cyclomatic complexity is the first complexity measurement that utilizes the graph theory conceptualization [53]. It measures the complexity based on the number of pathways a program can take through a decision tree. Halstead’s volume measure of complexity considers how much information is required to describe a system’s parts and interfaces [54]. Hölttä and Otto proposed interface complexity that measures complexity based on how much a change in one component can affect another through its interfaces. It can be utilized to minimize the impact of changes in rework time [55]. Summers and Shah developed a coupling complexity theory based on how much a system can be decomposed. It can be applied to assess the system’s difficulty, compare the system, and help with modularization and decomposition efforts [40]. Sinha et al.’s structural complexity quantifies complexity based on how dense and centralized the structure is and the difficulties of component and interface development. It can be utilized to predict the system and subsystem costs and support modularization [56,57]. McComb et al. [31] explored how leveraging the properties of configuration design problems can inform design team characteristics, such as team size and interaction frequency. Broniatowski and Moses introduced a descriptive complexity measurement, which considers how much information is required to describe the system’s structure. The system’s architecture can be abstracted with modularization so that the system can be understood with less cognitive effort [58].

Previous approaches to complexity measurement only measured the system complexity as a whole and did not consider the task constraint during calculation. Therefore, they fail to predict the optimal team size of a complex system such as the SOS. Developing a new complexity measurement that can address such an issue is crucial. Such areas often need to be included in the literature and are the focus of this article.

## 3 Task Constraints and Effective Entropy

This work started from two basic ideas. First, we assume Ashby’s law of requisite variety [2] holds for all complex

systems: a successful complex system must have the required level of variety (or complexity, roughly) of the target tasks; too little makes the system infeasible, and too much renders it inefficient. Second, suppose successful RL can be devised for a complex system to acquire its capabilities by itself [7,8]. In that case, the system naturally attains its level of complexity as required by the training tasks.

These assumptions imply that if multiple agents can explore more system states during training in a multiagent RL environment, they have a greater potential to complete the task. Information entropy is a common practice for measuring various states [59]. It can be used to gauge how much choice is involved in selecting an event or how uncertain the outcome is. Given  $p_i$  represents the probability of a system event or action state happening, the total information entropy or complexity of the system is just the negative of the sum of the probability times log probability of each possible event, as expressed with Eq. (1):

$$\text{Information Entropy} = - \sum_i^N p_i \log p_i \quad (1)$$

where the sum of  $p_i$  is equal to 1 and  $N$  is the number of possible states.

The principle of maximum entropy states that, when choosing a probability distribution to model a set of data or to represent a state of knowledge, one should select the distribution with the highest entropy among all those that satisfy the given constraints. This choice was made under the assumption that it represents the least bias and does not assume any additional information that is not explicitly provided. When no constraints are involved, Eq. (1) predicts that more possible system states lead to higher information entropy, and if a system has a fixed number of states, the information entropy is maximum if the probability of every state is equal. However, when task constraints are present, additional information becomes available to the system during state exploration, and Eq. (1) is no longer fully applicable. If such task constraints can be given and testable, the Lagrange multipliers method can be applied to entropy calculation. In many engineering cases, acquiring the knowledge of such constraints can be challenging, and incorrect constraints can lead to inaccurate inferences.

**3.1 Task Constraints.** Increasing the number of agents in multiagent RL training adds total information entropy to the system as more states and team actions can be explored. For example, suppose we compare pushing a rectangular box with 10 and 5 identical agents. The ten-agent team could move the box much faster than the five-agent team. Similarly, if we use these two teams to rotate the box, the ten-agent team can exert a maximum torque double that of the five-agent team; therefore, within a unit of time, the ten-agent team will be able to turn the box with a much larger maximum degree compared to the five-agent team.

*Task constraints:* Note that the usefulness of such possible added speed or turning degrees depends on the constraints associated with the task and its environment, which can be called *task constraints*, denoted as  $C_T$ , and may hinder or restrict the effect of agent team actions. When a team of agents works collectively in a task environment,  $T$ , constrained by  $C_T$ , then the constraints will render the team action space  $A$  into two subsets:

$$A = A^e \cup A^i \quad (2)$$

where  $A^e$  represents team actions that are effective given the task constraints and  $A^i$  represents team actions that are no longer effective given the task constraints.

An example of a task constraint is the *space constraint*, which limits the maximum number of agents participating in the task. If space can accommodate only five robots, then the other five

robots in a ten-robot team will be *prevented* from acting *entirely*, although their action potential can still be counted in standard information entropy calculation. In the case of box-pushing, if there are obstacles, then the agents must be conscientious not to over-push the box to the obstacles.

We categorize task constraints into two main types: “environmental constraints” and “agent-related constraints.” Environmental constraints refer to the limitations on resources, both *physical* and *informational*, that the task environment imposes on the actions of the agent team. An example of this is the physical space constraint mentioned previously. On the other hand, agent-related constraints define the limitations on the functional *capabilities* and temporal *availability* of the agents’ actions and interactions. For instance, in scenarios where human designers act as agents, their cognitive limitations regarding actions and interactions are considered agent-related constraints.

Distinguishing between task constraints and problem constraints is crucial. Problem constraints define the parameters for acceptable solutions, whereas task constraints affect the methodology for achieving those solutions. In many engineering optimization problems, problem constraints are integral to the problem’s definition. Conversely, task constraints emerge progressively as the team of agents is formed, highlighting agent-related constraints and as the solution methodology is developed or executed, revealing environmental constraints. We contend that the observed limitations on entropy increase accompanying the team’s expansion are predominantly attributable to the effects of task constraints.

*Team action threshold:* In complex task environments, pinpointing and quantifying the impact of task constraints can be challenging. Shifting our focus to the influence of these constraints, we examine them through the lens of actions taken by agent teams, which can be simulated to some extent. Generally, the potential impact of team actions tends to increase with team size. However, when task constraints are present, they may limit specific actions of the agent team. Consider if the metrics for evaluating actions are defined in terms of “amount,” “strength,” or “coverage” relative to team size. A cap on these metrics can be termed a *team action threshold*, distinguishing between effective actions  $A^e$  and ineffective actions  $A^i$ . With specific task constraints  $C_T$  in place, increasing the team size boosts the number of effective actions  $A^e$  up to the team action threshold. Beyond this threshold, further increases in team size become counterproductive due to task constraints, resulting in a rise in ineffective actions  $A^i$ . For instance, if a ten-agent team attempts to rotate a box, a single team action might achieve a maximum of 50 deg rotation, assuming each agent contributes to a 5-deg rotation. Yet, with task constraints  $C_T$  encompassing *Narrow Pathways* and *Obstacles*, any team action resulting in more than a 20-deg rotation becomes ineffective, as it would cause collisions with obstacles or pathway borders. In this context, the 20-deg rotation defines the *team action threshold*.

The concept of the team action threshold plays a significant role in multiagent RL. It is crucial to highlight that, within a team, agents can autonomously discover the team action threshold through a process of trial and error inherent in multiagent RL. In successful instances of multiagent RL, teams initially experiment with a range of potential actions as part of the exploration phase. As time progresses, they learn to disregard actions that fail to yield the anticipated rewards, refining their strategy to focus solely on actions that are effective. In the context of the earlier box-pushing scenario, ineffective actions that cause collisions with walls or obstacles would result in negative rewards. Assuming it is possible to represent the team actions graphically, the output from teams that successfully learn should only include effective actions  $A^e$  and exclude the ineffective ones  $A^i$  that surpass the team action threshold. Such a graphical representation illustrates the team’s ability to optimize its actions within the constraints imposed by the task constraints, showcasing the adaptability and efficiency of multiagent RL systems.



**3.2 Effective Entropy as a Team Complexity Measure.** Recognizing task constraints as a significant factor in restricting effective team actions has led us to introduce an *effective information entropy* or *effective entropy* for short. For a given task, task constraints, and a team of agents, there are corresponding effective team actions  $A^e$  and ineffective team actions  $A^i$ . Only effective team actions  $A^e$  should be considered for system complexity evaluation. Assuming  $A^e = \{a_1^e, a_2^e, \dots, a_M^e\}$ , we have:

$$\text{Effective entropy} = - \sum_{i=1}^M p(a_i^e) \cdot \log_2(p(a_i^e)), \quad (3)$$

$$\text{where } a_i^e \in A^e, \quad M = |A^e|$$

As mentioned earlier, for a specific task, when the team size increases, the effective entropy should increase as more possible states can be explored and  $A^e$  becomes bigger. However, because there is a *team action threshold* determined by the task constraints, the effective entropy will decrease as the sum of  $p_i$  within the team action threshold decreases due to the increase in the number of ineffective actions  $|A^i|$ . Therefore, the “best team size” happens when the “maximum effective entropy” is attained or the “team action threshold” is identified.

There can be ways to identify  $a_i^e \in A^e$ . If one can predefine task constraints and apply these constraints to exclude ineffective team actions  $A^i$ , it is possible to compute the effective entropy based on the identified  $A^e$ . However, in the cases of self-organizing systems working in dynamic environments where the constraints may not be definable, such simple methods become inapplicable. We propose a multiagent RL method to tackle this problem, which does not require explicit prior knowledge of task constraints and the dynamics of the changing environment.

**3.3 Computational Cost.** Because the best team size of an agent team depends on its team action threshold, we focus on identifying the latter. Our approach is RL based: *agents in a team can identify their team action threshold through trial and error during multiagent RL processes*. However, the issue with this method is that the training *simulation runs* needed to determine the team action threshold require enormous computing time.

For a given agent team of a certain size, it takes  $\text{SimRuns} = \text{Episodes} \times \text{RunsPerEpisode}$  for training to obtain a successful learning team of that size. If there are different numbers of team sizes,  $\text{TeamSizeNum}$ , then the total simulation runs needed will be

$$\text{Total Sim Runs} = \text{Team Size Num} \times (\text{Episodes} \times \text{Runs Per Episode}) \quad (4)$$

Consider a team with ten numbers of size (e.g., size = 2, 3, ..., 11), the number of episodes is 40,000, and it takes 100 runs per episode; the total training simulation runs will be  $10 \times 100 \times 40,000 = 40,000,000$ . If one *OneEpisodeRun* takes 0.5 s on a powerful PC, the *total simulation time* can be 33 weeks. In this article, we propose a method to predict the best agent team size by

identifying the *team action threshold* that requires only a fraction of the total runs indicated in Eq. (4). *Effective learning region* and *action maps* are essential instruments in this method.

**3.4 Effective Learning Region and Action Map.** For a given task environment,  $E_t$ , we consider  $E_t = \{r_1, r_2, \dots, r_R\}$ , i.e., the task environment is composed of different regions (or aspects),  $r_1, r_2, \dots$ . Each region  $r_i$  of the task may have its own task constraint situations and a unique team action threshold.

From a multiagent RL perspective, not all regions require extensive learning. A particular region may exhibit much more extensive learning than others due to its highly constrained situation. In the aforementioned box-pushing example, if the pathway changes from *wide-open* to *narrow-down*, it is conceivable that the agents need to learn how to coordinate well to rotate the box in line with the narrow pathway before moving into it. In this case, intensive learning happens around the *region* of the narrow pathway entrance, where the agents learn what to do (i.e.,  $A^e$ ) and what to avoid (i.e.,  $A^i$ ). We call such a region an *effective learning region*, which is the place where the critical *action threshold* can be identified.

Given an effective learning region, the next question is how to identify its action threshold. For this purpose, we propose an instrument called a *team action map*, or *action map* for short, defined by plotting (or visualizing or simply counting) team actions of a *successfully trained* agent team in the given effective learning region. Since agents in a successfully trained team have already learned to avoid ineffective actions  $A^i$  beyond the team action threshold, the map should be blank if one attempts to plot the “beyond-threshold” actions  $A^i$ . Plotting *action maps* of the team of the maximum possible size and then identifying the action threshold as diminishing of actions can significantly reduce computing requirements.

## 4 Computational Method

Based on the effective entropy measure and the concepts introduced earlier, a workflow for computationally predicting the optimal team size of SOS is shown in Fig. 1.

The first step is to train an agent team of a large size (e.g., ten agents) with multiagent RL using different random seeds (e.g., 100 random seeds). With such a large team size, we can generate a greater variety of team actions for later visualization and analysis. Then we run the successfully trained neural networks *in greedy* and plot the *action maps* by plotting the center position of the box for different rotating team actions. On the basis of the maps, we can determine the max *team action threshold* in the identified *effective learning region*. Meanwhile, the *probability distribution* of team actions can be simulated using Monte Carlo simulation. Next, the *effective entropy* can be calculated for the different numbers of agents, and the *optimal team size* of the SOS team can be identified.

From this workflow, it can be seen that the multiagent RL training is carried out for only the largest team size. Therefore, the total

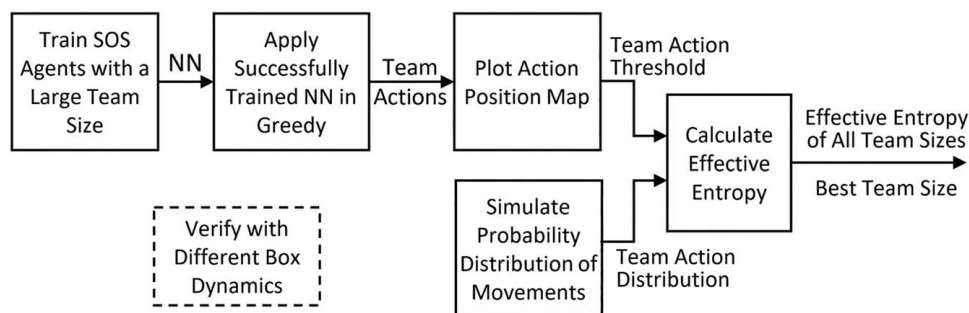


Fig. 1 A computing workflow of optimal team size prediction of SOS

simulation runs for training are reduced to:

$$Total\ Sim\ Runs = 1 \times (Runs\ Per\ Episode \times Episodes) \quad (5)$$

which is only a fraction ( $1/TeamSizeNum$ ) of the original simulation runs in Eq. (4).

For this research, we further verified the proposed model by running simulations with different box dynamics, as indicated by the dash-lined box in Fig. 1.

**4.1 Multiagent Reinforcement Learning.** Multiagent RL is the first component of our computational method. Using the box-pushing case of our previous work [8] for explanation, Fig. 2 shows three kinds of box movements: along the box's  $x$ -axis,  $y$ -axis, and rotation. The agent team needs to push and rotate the box toward the goal area (the first row of Table 1). If there are no obstacles in the way, the complexity of the task is low. Adding obstacles increases the task complexity.

If one agent is in region 1 of the box, it will generate a one-unit box movement along the negative  $x$ -direction, a 0-unit movement along the  $y$ -direction, and a 1-unit rotation. Such box movement can be expressed with a vector:  $\langle x, y, \phi \rangle = \langle -1, 0, 1 \rangle$ . Similarly, we can get the movement vector of the box when an agent is in different regions of the box neighborhood:

- Region 1:  $\langle -1, 0, 1 \rangle$ , Region 2:  $\langle -1, 0, -1 \rangle$ , Region 3:  $\langle 1, 0, 1 \rangle$

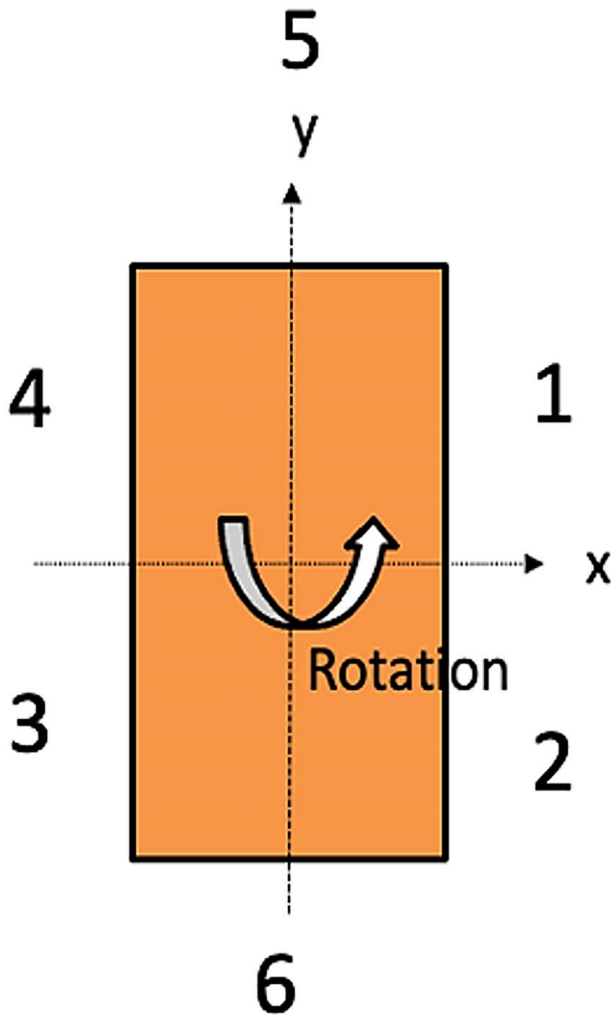


Fig. 2 Box-pushing task state representation with numbers indicating box regions

- Region 4:  $\langle 1, 0, -1 \rangle$ , Region 5:  $\langle 0, -1, 0 \rangle$ , Region 6:  $\langle 0, 1, 0 \rangle$

The multiagent RL details and the major parameters are shown in Table 1.

During the multiagent RL training process, agents randomly explore their action space. They can be in different positions in the box neighborhood. The probability of each agent in any of the six regions is equally likely, and the total box movement is just the sum of the vectors generated by each agent.

**4.2 Effective Entropy and Optimal Size Prediction—A Case Study.** The complexity of the box-pushing task mainly comes from the box's rotation to avoid obstacles [8], as moving the box along its  $x$ - or  $y$ -axis is less complex. Therefore, the box rotation is chosen as the focused team action. The unit rotation is defined as 5 deg, meaning each agent can contribute 5 (–5) deg of box rotation if positioned in regions 1 or 3 (2 or 4) and 0 deg if in regions 5 or 6. Monte Carlo simulations can be applied to generate the probability distribution of each possible rotation movement, as shown in Fig. 3(a).

**Obstacle constraints:** Based on Eq. (1), the resulting total entropy of each agent team size is shown in Fig. 3(b), which does not consider constraints; the total entropy of the system increases along with the team size and cannot predict the optimal team size of SOS given task constraints. Applying task constraints to the effective entropy calculation means identifying the rotation action threshold posed by the obstacle and the walls.

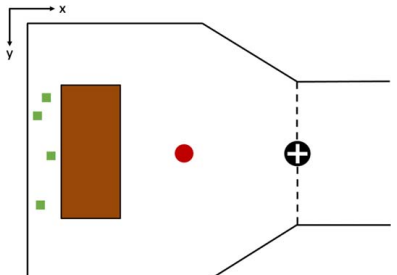
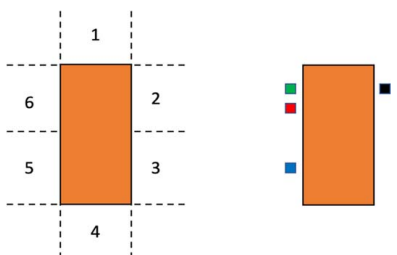
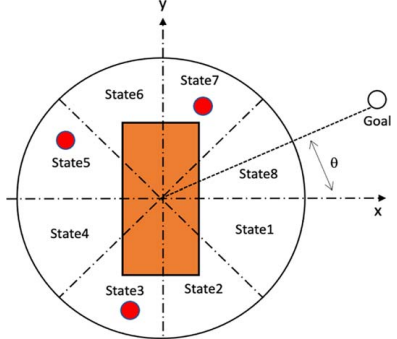
**Rotation threshold:** Figure 4 shows the most constrained region in the box-pushing environment. The circle in the middle represents an obstacle. The dashed arrows pointing from the obstacle vertically to the side walls are the narrowest paths along the box-pushing trajectory, signifying the bottleneck of the box-pushing task. The regions of the solid arrows pointing to the goal dot with “+” are considered the effective learning region, where the maximum rotation team action will likely occur.

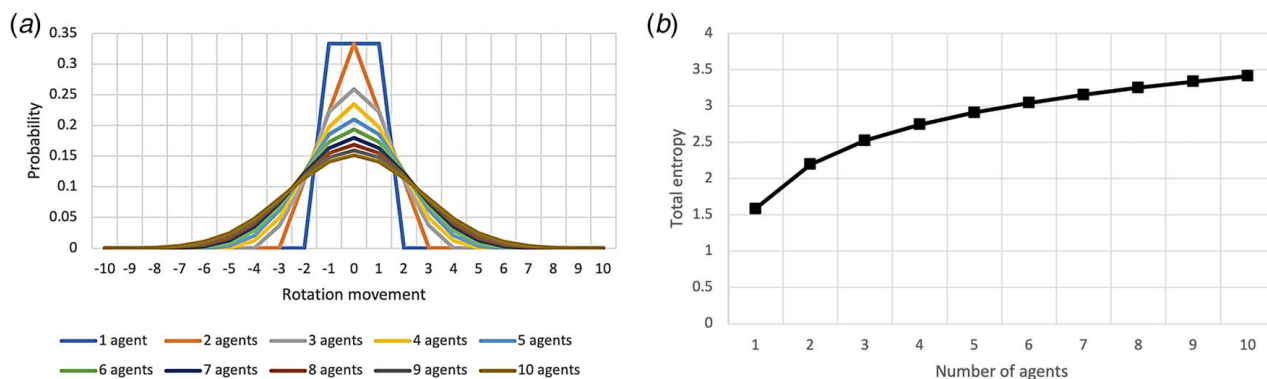
**Rotation maps:** Following the workflow in Fig. 1, we first train the TeamSize = 10 team with 100 random seeds [8], then we apply the successfully trained neural networks in greedy, and plot rotation maps of the box, as shown in Fig. 5. Each dot in a plot of a specific degree shows the center position of the box where the rotation action of that particular degree happened. The rotation maps combine all successful box-pushing runs and separate them based on rotation degrees. For example, for the 5-deg rotation Fig. 5 (5 deg), the box is pushed to rotate 5 deg with different dots indicating different 5-deg events in the same or different simulation runs. Figure 5 shows that as the rotation degree increases, the number of box position points decreases and gradually shifts from the right (more constrained region) to the left (less constrained region). When rotations are greater than 15 deg, they rarely happen and hence are missing for box-pushing. They, thus, should be excluded from the effective entropy calculation. Therefore, 15 deg of rotation is the rotation threshold for effective entropy calculation.

To illustrate how the team rotation threshold may change with different team sizes, we plotted rotation maps of TeamSize = 4 (Fig. 6) and TeamSize = 20 (Fig. 9). Figure 9 (15 deg) shows the box position at the final stage of box-pushing before the finish line. In Fig. 9 (20 deg), however, few points appear inside the effective learning region, indicating 15 deg of rotation should be considered as the rotation threshold for TeamSize = 10.

It is worth emphasizing that although the team action maps, as illustrated in Figs. 5 and 6, aid in visualizing the action threshold, multiagent RL training is the critical process that did the action threshold identification. The action map-based visualization can introduce subjective bias due to the potential ambiguity of the threshold in the maps. Alternatively, quantifying action occurrences beyond the effective learning region offers a more objective method. By establishing a numerical threshold, actions that occur

**Table 1 Major parameters of the box-pushing case study (see Ref. [8] for details)**

Parameters	Illustration or equation	Description
The box-pushing problem		<i>Task:</i> The solid rectangular box is supposed to be moved into the goal position marked with "+". <i>Agents:</i> The green squares have limited sensing and pushing capabilities to push and rotate the box. <i>Constraints:</i> The center dot obstacle and the side walls. The box cannot hit the obstacle or side walls.
Box neighborhood		<i>Regions and step:</i> An individual agent can stay in, or move to, one of the six regions of the box neighborhood. A region can have multiple agents. A move from a region to one of its neighboring regions is defined as one step of the move.
RL state space and action space		<i>State space:</i> The task state space is defined by dividing the box's vicinity into eight segments. In addition, the box's orientation angle $\beta$ is also taken as a state variable. $S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9\}$ . For $s_1$ through $s_8$ , if an obstacle is present, the value is 1; otherwise, it is 0. For $s_9$ , the value range is set to be $[-1, 1]$ to cover 0 to 360 deg for easier training. <i>Action space:</i> At each time-step, an agent can choose a place in one of the six regions to push the box, indicated as $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ .
RL reward schema	$R_{\text{total}} = w_1 R_{\text{dis}} + w_2 R_{\text{rot}} + w_3 R_{\text{col}} + w_4 R_{\text{goal}}$	<i>Reward:</i> The total reward is a weighted summation of the reward items: distance, rotation, collision, and goal. All weights have fixed values [8].
Single-agent learning	Q-Learning: $Q_{t+1}(s, a) = E[r + \gamma \max_a Q_t(s', a')   s, a]$ DQN: Loss function: $L_t(\theta_i) = E \left[ \left( r + \gamma \max_a Q_t(s', a'; \theta_{i-1}) - Q_t(s, a; \theta_i) \right)^2 \right]$	<i>Deep Q-learning:</i> A DQN is used to approximate Q-table in Q-learning. <i>Value calculation:</i> Each agent $i$ has its own Q-value network $= Q_i(\theta_i)$ , which is updated through training by minimizing MSE loss $L_i(\theta_i)$ . <i>Action selection:</i> Each agent $i$ takes an $\epsilon$ -greedy policy to select its next action.
Multiagent learning	Common/shared information: $S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9\}$ $R_{\text{total}} = w_1 R_{\text{dis}} + w_2 R_{\text{rot}} + w_3 R_{\text{col}} + w_4 R_{\text{goal}}$ Individual/private information: Q-value network $= Q_i(\theta_i)$	<i>Multiagent deep Q-learning:</i> (1) Agents have the same sensing capability to sense the state of the task environment; (2) agents share the same reward function; (3) each agent learns its own neural network throughout the training process; and (4) each agent uses its own neural network in greedy to perform the box-pushing tasks after training.

**Fig. 3 Multiagent box-pushing task without considering constraints: (a) probability distribution of rotation versus agent team size and (b) total entropy versus agent team size**

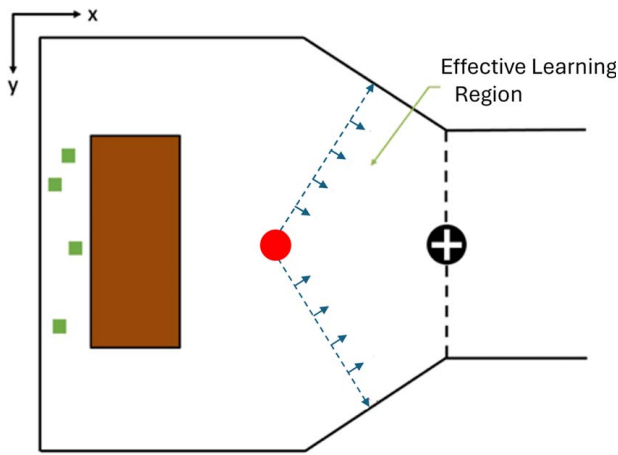


Fig. 4 Effective learning region of box-pushing task

less frequently than this set limit (e.g., specific degrees of rotation) can be objectively classified as reaching the team action threshold.

**Effective entropy and optimal agent team size:** Given the team rotation threshold = 15 deg, we can calculate the effective entropy. The maximum number of states for any agent team size is 7, i.e., -15, -10, -5, 0, 5, 10, and 15 deg of rotation. Each state's corresponding probability distribution is simulated using Monte Carlo simulation (Fig. 3(a)). For a small number of agents, e.g., two agents, the possible rotation states are -10, -5, 0, 5, and 10 deg, all inside the rotation threshold. So, effective entropy is the same as information entropy. For a larger team size of 10 agents, it has 21 possible states, from -50 to 50 deg. However, only seven states between -15 to 15 deg are effective. Thus, the effective entropy should only calculate the probability distributions up to these seven rotation states.

The plots of the effective entropy for different team sizes are shown in Fig. 7, which indicates that five is the optimal team size for the box-pushing case. The result is consistent with the performance score of our previous box-pushing case study [8], shown

in Fig. 8. The performance score in Fig. 8 is the percentage of successful simulation runs out of 100 simulations with random initial agent positions.

Figure 7 indicates that as the number of agents increases, the effective entropy increases until the team size is 5 because more agents add more team actions to the system. The entropy peaks around TeamSize = 5 and then diminishes as adding agents only leads to ineffective team actions. The effective entropy calculation can help us mathematically explain why the “magic” team size is 5 in our box-pushing case study: at this team size, the system complexity, measured in our effective entropy, is at its peak and matches the complexity of the task, rendering the system achieving the best performance [60].

**4.3 Verification.** To verify the effective entropy measure, we carried out another box-pushing experiment that uses weak agents. The simulation environment is the same except that each agent can rotate only 2.5 deg, instead of 5 deg, for each unit push action. With the weak agents, it can be imagined that the optimal team size should be around 10 agents, as the system needs twice the size of the original agents. Following the same steps of the previous case: (1) the agents are trained with multiagent RL with a large team size of 20 agents, (2) the successfully trained agents are applied in greedy to box-pushing, (3) rotation maps of the 20 agents are plotted, (5) the rotation threshold is determined, and (6) the probability distributions are calculated using Monte Carlo simulations by considering the rotation threshold.

The rotation maps of 20 agents with different rotation degrees are shown in Fig. 9, which indicates that 10 deg of rotation should be the rotation threshold, as box centers rarely pass the effective learning region beyond this degree. Note that 10 deg of rotation is slightly smaller than the 15 deg of rotation threshold determined in the previous case. As the agents' pushing strength is only half that of the original agents, it can generate more combinations of movements during training. Therefore, the rotation threshold can be smaller when agents learn from a greater variety of actions.

The probability distribution of each rotation degree can be simulated using Monte Carlo simulations as in the previous box-pushing case. After applying the rotation threshold = 10 deg, we can

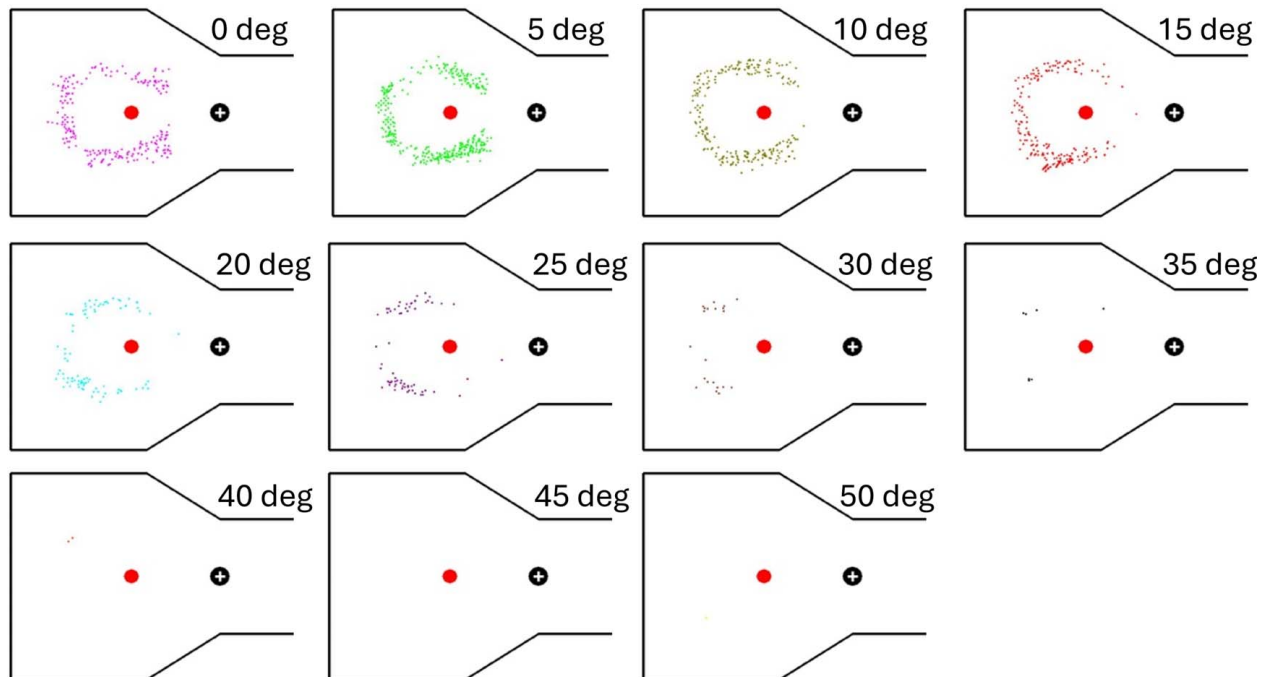


Fig. 5 Action maps of different rotation movements with ten agents



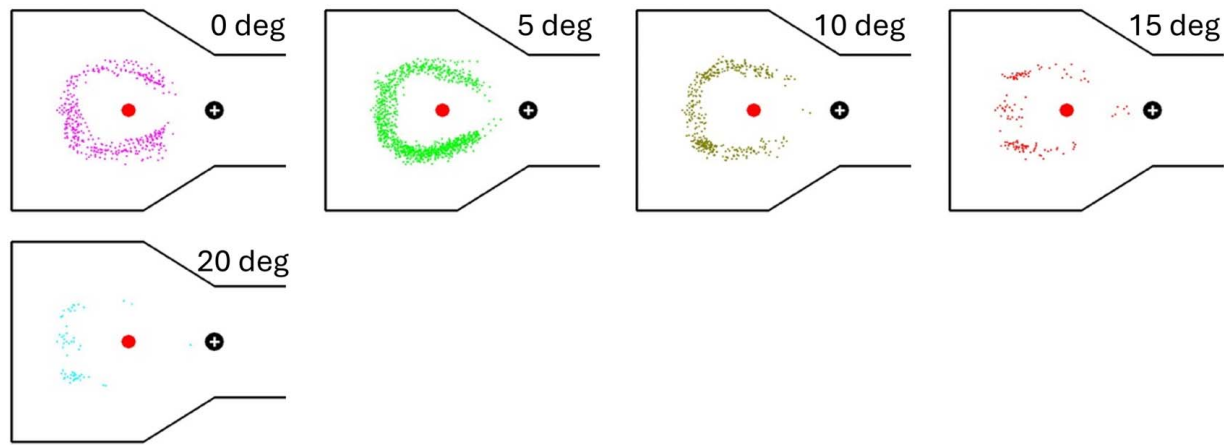


Fig. 6 Action maps of different rotation movements with four agents

calculate the effective entropy with different team sizes, as shown in Fig. 10. The effective entropy first increases up to a point and then declines, with eight and nine agents obtaining the maximum effective entropy.

Figure 11 shows the results of the learned agent teams performing box-pushing in greedy. The performance score increases to ten agents; after 10, the performance scores stay close to 90. This verifies that the effective entropy can accurately predict the optimal agent team size.

The performance scores are quite different across team sizes in the previous case (Fig. 8). As depicted in Fig. 11, however, the score increases and stays about the same. The reason is that in the weak-agent box-pushing case, each agent's pushing capability is only half of the strong agents in the previous case. Finer agent action compositions allow the system to be more robust in avoiding obstacles, as adding weak agents can make a greater variety of small movements, resulting in higher performance scores than strong-agent teams.

It is worth noting that the weak agents exemplify the "agent-related constraints," a type of task constraint defined in Sec. 3. The change in agent's capability to 50% of the previous case significantly impacted the optimal size of the agent team, shifting it from five agents to ten agents. Although, in this case, the effect is straightforward, the outcomes affirm that the proposed effective entropy measure and the method described earlier can be practical tools for predicting the optimal agent team size under both environmental and agent-related task constraints.

## 5 Discussion

*Two principles:* The approach to effective entropy described earlier is based on two principles. First, Ashby's law of requisite

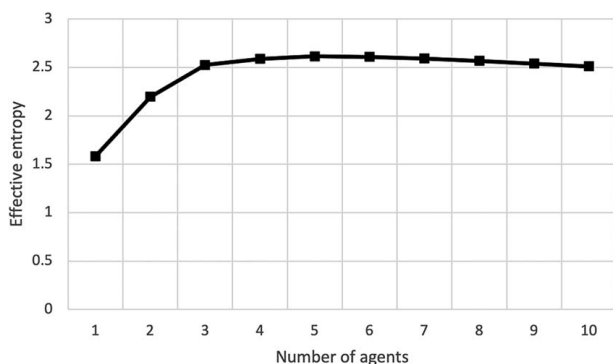


Fig. 7 Effective entropy with varying numbers of agents in the box-pushing task

variety posits that for any task within a specific environment characterized by a particular degree of variety or complexity, the corresponding system must exhibit an equal or greater level of complexity to ensure stable or successful task completion [2]. In the context of RL, when agents are trained to develop self-organizing capabilities for predefined tasks, the emergent self-organizing mechanisms, namely, the trained neural networks, can be deemed to have attained an adequate level of system complexity that mirrors the intricacy of the task at hand. Second, we consider the principle of maximum entropy [61], which suggests that estimating a probability distribution should favor the one with the highest residual uncertainty, which aligns with known constraints, thereby maximizing entropy. While analytical methods, e.g., the Lagrange multiplier technique, exist to compute maximum entropy, they often become untenable due to a lack of prior information regarding the task constraints. To circumvent these limitations, this article introduces a data-driven methodology for approximating the maximum entropy for teams of self-organizing agents and mapping the size of agent teams with reduced computational requirements. This method has been substantiated through application to a relatively straightforward self-organizing system task, demonstrating its viability and effectiveness.

*Critical steps and concepts:* Our method comprises a sequence of crucial steps and concepts. First, it necessitates identifying a select few "pivotal team actions" with evaluative action metrics (e.g., team size associated with the team action), thereby sharpening the focus of the analysis. These pivotal team actions often correlate with potential impediments, or bottlenecks, to task execution, which can emerge during multiagent RL procedures. In our case study,

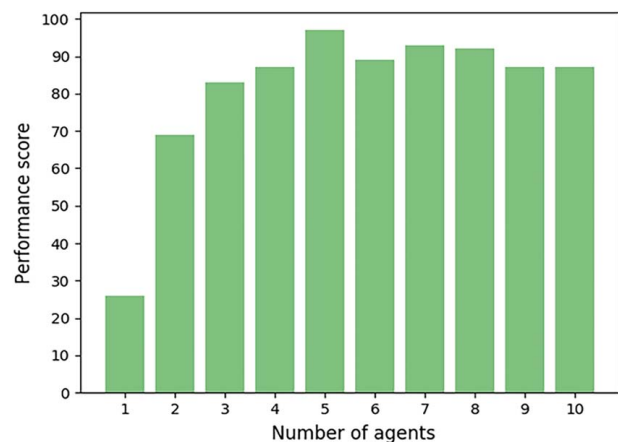


Fig. 8 Performance score with different numbers of agents in the box-pushing task [8]



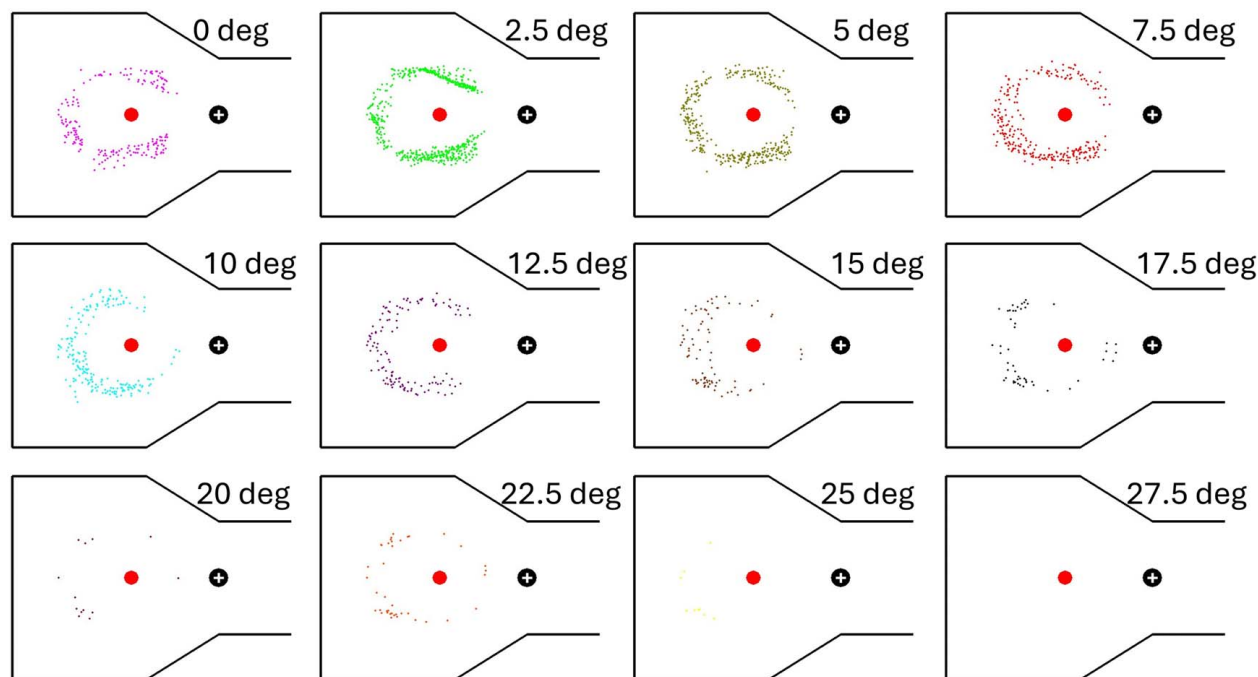


Fig. 9 Action maps of different rotation movements with 20 agents

the pivotal team action, determined from our prior multiagent RL research [8], was the “rotation of the box.” Second, it is imperative to identify environmental and agent-related factors as “task constraints” that influence the execution and effect of the pivotal team actions. The obstacles and walls shown in Fig. 4 are task constraints in our case study. Third, one must investigate the critical limits of the pivotal team action as the “team action threshold”—essentially, the point at which team performance begins to deteriorate significantly due to the influence of task constraints. From an effective entropy computation point of view, the action threshold is why the effective entropy can attain its maximum value. In our approach, we rely on multiagent RL training to identify these critical limits, where agents are conditioned to avoid surpassing these bounds in their collective actions. Plotting *action maps* of large-size trained teams can help illustrate the action threshold. Last, the “evaluative action metric” that relates to the agent team configurations (e.g., team size) at the “team action threshold” is used for computing effective entropy, which, in turn, informs the optimal size of agent teams.

**Homogeneity:** It should be noted that the agent teams evaluated in this study are homogeneous; all agents are designed with the sole action of “push-box,” even though they exhibit varied

behaviors in choosing which region to push due to the diversity in their learned neural networks. In scenarios involving heterogeneous agent teams, where agents can perform different actions, the parameters of effective entropy assessment differ. Instead of having “team size” as the variable on the sole  $x$ -axis, one would need to include multiple variables, such as “agent-type1-size,” “agent-type2-size,” and so forth, to account for the variety in agent functions. Consequently, the graphical representation of effective entropy would not simply be a curve, as shown in Fig. 7, but rather a multidimensional surface. The apex of this surface would not only define the optimal team composition but also reflect the intricate interplay of agent heterogeneity. It is expected that even in heterogeneous cases, it is still critical to closely consider the “pivotal team actions,” “task constraints,” “team action threshold,” and “team action metrics” along different agent function dimensions and the interactions between them. Further study is needed to explore this direction.

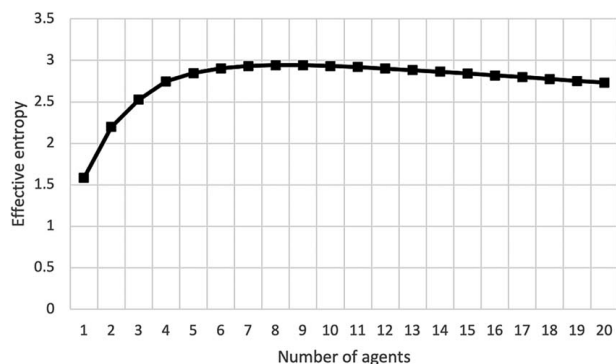


Fig. 10 Effective entropy with varying numbers of agents in 20-agent box-pushing

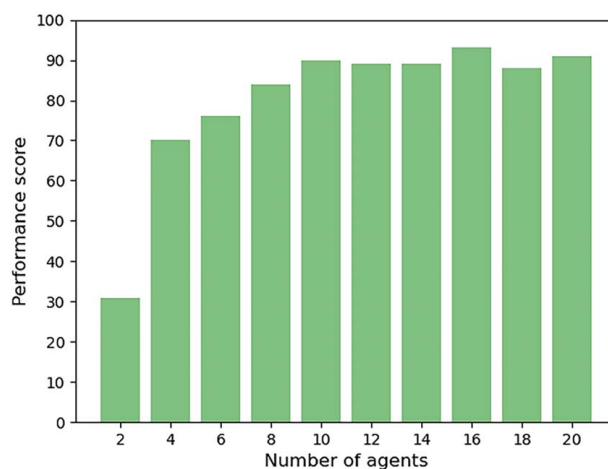


Fig. 11 Performance score with different numbers of agents in heaving box-pushing

**Design problems:** Engineering design problems can be much more complex than the box-pushing task, primarily due to intricate relations among various design parameters and, hence, complex interactions among team members. Nevertheless, the approach of this article can provide valuable insights for dealing with such problems. McComb et al. [31] explored how leveraging the properties of configuration design problems can inform design team characteristics, such as team size and interaction frequency. They demonstrated that the optimal team size depends negatively on the alignment of objectives and positively on the roughness of local and global design space. The result is consistent with Ashby's law of requisite variety and provides quantitative measures. It is conceivable that with more diverse objectives and rougher design spaces, the problem becomes more complex, demanding more variety of team actions and, hence, more members. However, when task constraints are considered, the team will not be "the larger, the better." In human design teams, physical and cognitive factors that limit interactions can be agent-related "task constraints," and the interaction can be treated as the "pivotal team action." As shown in Fig. 4(b) of Ref. [31], when the interaction frequency increases after the "action threshold," the optimal team size decreases. Furthermore, the lack of resources can be considered "task constraints," as the authors found the resource has significant contributions to their model accuracy and "when more resources are available, it is beneficial to increase team size, spreading resources among a greater number of individuals to increase the extent to which work can be completed concurrently."

In a separate study, Hulse et al. [32] applied an adapted multiagent reinforcement learning method as an optimization algorithm to solving a quadcopter design problem, with each agent in charge of designing one component of the overall system. In addition to the advantages of their distributed multiagent approach over other centralized optimization algorithms, the authors found the positive effect of synchronization on the design performance. In this case, providing synchronized value feedback to a higher level can be considered a "pivotal team action," and organizational or physical factors that hinder synchronization signify the "task constraints." If such relevant factors cannot be identified and removed, an "action threshold" can exist to determine the team performance level that is inferior to the desired.

## 6 Conclusions and Future Work

This article introduces an effective entropy measure for evaluating system complexity and devises a predictive framework to determine the optimal size of self-organizing agent teams. The method involves orders of magnitude fewer multiagent reinforcement learning simulations. It focuses on "pivotal team actions" and facilitates assessing the impact of "task constraints" and determining the "action threshold" through multiagent RL training. The "evaluative metric" of the agent team configuration at the action threshold is applied to compute the effective entropy of the agent team, informing the optimal agent team size. Through the lens of entropy, this approach enhances the understanding of system complexity and provides a practical method for optimizing team configurations in self-organizing systems. Through a box-pushing case study, we demonstrated how each key concept and step is applied and carried out with fewer multiagent RL training simulations. The following are the conclusions drawn from the results and discussions described earlier.

- The key factor in determining the optimal size for an agent team lies in task constraints that create agent team action thresholds. Although problem constraints and properties do influence team configuration, it is through task constraints—be they environmental, agent-related, or both—that their impact becomes apparent.
- Understanding and analyzing task constraints requires attention to several critical concepts, including identifying pivotal team actions to focus analysis, recognizing environmental and agent-related factors that serve as task constraints, and

identifying team action thresholds that distinguish effective actions from ineffective ones. In addition, visualizing these thresholds through action maps can be effective when practical.

- Multiagent RL training is a critical method in our approach to identifying the impact of task constraints as agent team action thresholds. The procedure involves training the agent team at a large team size, simulating the trained team in greedy for the task, identifying the team action threshold, and calculating the effective entropy across different team sizes. The optimal team size is the one that maximizes the effective entropy.
- The proposed method significantly lowers the computational resource demands by a factor equal to the range of agent team sizes under consideration.

From an engineering point of view, the potential applications of our findings unfold in two primary directions. The first revolves around leveraging our insights into task constraints and pivotal team actions. The concept of effective entropy can be employed to gauge the complexity of engineering design challenges. By assessing task constraints and pinpointing pivotal team actions via "thought-based" or "simulation-based" approaches, it becomes feasible to ascertain the optimal team size that aligns with the targeted complexity level. This insight aids in refining team configurations, thus enhancing the efficiency and effectiveness of the design processes. Moreover, incorporating task constraints, like resource limitations, into the effective entropy calculation facilitates the determination of the most suitable team size and resource distribution strategy. This, in turn, optimizes resource utilization and boosts system performance. The second application domain is computational, focusing on identifying crucial team actions and deploying the proposed multiagent RL-based simulation method to discern the team action threshold. This approach allows designing multifaceted multiagent or self-organizing systems by utilizing our method to establish optimal team configurations.

The simplicity of the box-pushing task was instrumental in demonstrating the principles and procedures of the proposed method. However, this case study's narrow focus highlights the limitations of our research, particularly when considering the complexities of tasks such as assembling intricate structures, which remain unaddressed. In addition, the uniformity of the agents in the box-pushing case does not account for challenges such as functional composition found in teams with diverse capabilities. A further limitation is the study's emphasis on physical actions; as tasks become more knowledge-intensive and less physically oriented, new methods must be devised to extend the applicability of our approach to encompass a broader range of activities.

Our future research aims to further test the proposed effective complexity measurement and optimal team size prediction method across a broader spectrum of complex tasks. This effort will involve several key areas of expansion: enriching the taxonomy of task constraints, developing strategies for teams of heterogeneous agents, and investigating frameworks for self-organizing problem solvers that incorporate elements of knowledge work and decision-making alongside physical actions.

## Acknowledgment

This article was based on the work supported in part by MTI Co. Ltd. and Nippon Yusen Kaisha (NYK). The authors thank the sponsors and the MTI team for their discussions and insights on this research.

## Conflict of Interest

There are no conflicts of interest.

## Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

## References

- [1] Reynolds, C. W., 1987, "Flocks, Herds and Schools: A Distributed Behavioral Model," *ACM SIGGRAPH Computer Graphics*, **21**(4), pp. 25–34.
- [2] Ashby, W. R., 1991, "Requisite Variety and Its Implications for the Control of Complex Systems," *Facets Syst. Sci.*, **7**, pp. 405–417.
- [3] Chiang, W., and Jin, Y., 2012, "Design of Cellular Self-Organizing Systems," *Intl. Design Engineering Technical Conferences*, Chicago, IL, Aug. 12–15, Vol. 45028, pp. 511–521.
- [4] Humann, J., Khani, N., and Jin, Y., 2014, "Evolutionary Computational Synthesis of Self-Organizing Systems," *AI EDAM*, **28**(3), pp. 259–275.
- [5] Khani, N., Humann, J., and Jin, Y., 2016, "Effect of Social Structuring in Self-Organizing Systems," *ASME J. Mech. Des.*, **138**(4), p. 041101.
- [6] Khani, N., and Jin, Y., 2015, "Dynamic Structuring in Cellular Self-Organizing Systems," *Design Computing and Cognition '14*, J. S. Gero and S. Hanna, eds., Springer, Cham, Switzerland, pp. 3–20.
- [7] Ji, H., and Jin, Y., 2018, "Modeling Trust in Self-Organizing Systems With Heterogeneity," *ASME 2018 Intl. Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Quebec City, Quebec, Canada, Aug. 26–29.
- [8] Ji, H., and Jin, Y., 2021, "Evaluating the Learning and Performance Characteristics of Self-Organizing Systems With Different Task Features," *AI EDAM*, **35**(4), pp. 404–422.
- [9] Ji, H., and Jin, Y., 2022, "Knowledge Acquisition of Self-Organizing Systems With Deep Multiagent Reinforcement Learning," *ASME J. Comput. Inf. Sci. Eng.*, **22**(2), p. 021010.
- [10] Dasgupta, P., 2008, "A Multiagent Swarming System for Distributed Automatic Target Recognition Using Unmanned Aerial Vehicles," *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans*, **38**(3), pp. 549–563.
- [11] Ruini, F., and Cangelosi, A., 2009, "Extending the Evolutionary Robotics Approach to Flying Machines: An Application to MAV Teams," *Neural Netw.*, **22**(5–6), pp. 812–821.
- [12] Lamont, G. B., Slear, J. N., and Melendez, K., 2007, "UAV Swarm Mission Planning and Routing Using Multi-Objective Evolutionary Algorithms," *2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, Honolulu, HI, Apr. 1–5.
- [13] Wei, Y., Madey, G. R., and Blake, M. B., 2013, "Agent-Based Simulation for UAV Swarm Mission Planning and Execution," *Proceedings of the Agent-Directed Simulation Symposium*, San Diego, CA, Apr. 7–10, pp. 1–8.
- [14] Chen, C., and Jin, Y., 2011, "A Behavior Based Approach to Cellular Self-Organizing Systems Design," *International Design Engineering Technical Conferences*, Washington, DC, Aug. 28–31, Vol. 54860, pp. 95–107.
- [15] Sutton, R. S., and Barto, A. G., 2018, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.
- [16] Busoni, L., Babuska, R., and De Schutter, B., 2008, "A Comprehensive Survey of Multiagent Reinforcement Learning," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, **38**(2), pp. 156–172.
- [17] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., and Vicente, R., 2017, "Multiagent Cooperation and Competition With Deep Reinforcement Learning," *PLoS One*, **12**(4), p. e0172395.
- [18] Tan, M., 1993, "Multiagent Reinforcement Learning: Independent vs. Cooperative Agents," *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, MA, July 27–29, pp. 330–337.
- [19] Watkins, C. J. C. H., 1989, "Learning From Delayed Rewards," *Ph.D. Dissertation*, Cambridge University, Cambridge, UK.
- [20] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Hassabis, D., et al., 2015, "Human-Level Control Through Deep Reinforcement Learning," *Nature*, **518**(7540), pp. 529–533.
- [21] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S., 2018, "Counterfactual Multiagent Policy Gradients," *Proceedings of the AAAI Conference on Artificial Intelligence*, Apr., Vol. 32, No. 1.
- [22] Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., and Whiteson, S., 2017, "Stabilising Experience Replay for Deep Multiagent Reinforcement Learning," *Intl. Conference on Machine Learning*, Sydney, Australia, Aug. 6–11, PMLR, pp. 1146–1155.
- [23] Hausknecht, M., and Stone, P., 2015, "Deep Recurrent q-Learning for Partially Observable MDPs," *2015 AAAI Fall Symposium*, Arlington, VA, Nov. 12–14.
- [24] Hochreiter, S., and Schmidhuber, J., 1997, "Long Short-Term Memory," *Neural Comput.*, **9**(8), pp. 1735–1780.
- [25] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y., 2014, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv preprint arXiv:1412.3555*.
- [26] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I., 2017, "Multiagent Actor-Critic for Mixed Cooperative-Competitive Environments," *31st Conference on Neural Information Processing Systems*, Long Beach, CA, Dec. 4–9.
- [27] Brown, N., and Sandholm, T., 2019, "Superhuman AI for Multiplayer Poker," *Science*, **365**(6456), pp. 885–890.
- [28] Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., and Mordatch, I., 2019, "Emergent Tool Use From Multiagent Autocurricula," *The 8th International Conference on Learning Representations*, Virtual Conference, Apr. 26–May 1.
- [29] Wu, S. A., Wang, R. E., Evans, J. A., Tenenbaum, J. B., Parkes, D. C., and Kleiman-Weiner, M., 2021, "Too Many Cooks: Bayesian Inference for Coordinating Multiagent Collaboration," *Top. Cogn. Sci.*, **13**(2), pp. 414–432.
- [30] Bar-Yam, Y., 2002, *General Features of Complex Systems*, *Encyclopedia of Life Support Systems (EOLSS)*, UNESCO, EOLSS Publishers, Oxford, UK.
- [31] McComb, C., Cagan, J., and Kotovsky, K., 2017, "Optimizing Design Teams Based on Problem Properties: Computational Team Simulations and an Applied Empirical Test," *ASME J. Mech. Des.*, **139**(4), p. 041101.
- [32] Hulse, D., Tumer, K., Hoyle, C., and Tumer, I., 2019, "Modeling Multidisciplinary Design With Multiagent Learning," *Artif. Intell. Eng. Des. Anal. Manuf.*, **33**(1), pp. 85–99.
- [33] Chen, L., and Li, S., 2005, "Analysis of Decomposability and Complexity for Design Problems in the Context of Decomposition," *ASME J. Mech. Des.*, **127**(4), pp. 545–557.
- [34] Joshua, D., Summers, J. D., and Shah, J., 2010, "Mechanical Engineering Design Complexity Metrics: Size, Coupling, and Solvability," *ASME J. Mech. Des.*, **132**(2), p. 021004.
- [35] Allaire, D., He, Q., Deyst, J., and Willcox, K., 2012, "An Information-Theoretic Metric of System Complexity With Application to Engineering System Design," *ASME J. Mech. Des.*, **134**(10), p. 100906.
- [36] Chen, W., Fuge, M., and Chazan, J., "Design Manifolds Capture the Intrinsic Complexity and Dimension of Design Spaces," *ASME J. Mech. Des.*, **139**(5), p. 051102.
- [37] Simon, H. A., 1962, "The Architecture of Complexity," *Proc. Am. Philos. Soc.*, **106**(6), pp. 467–482.
- [38] Bashir, H. A., and Thomson, V., 1999, "Estimating Design Complexity," *J. Eng. Des.*, **10**(3), pp. 247–257.
- [39] Sheard, S. A., and Mostashari, A., 2010, "A Complexity Typology for Systems Engineering," *INCOSE International Symposium*, Chicago, IL, July 11–15, Vol. 20, No. 1, pp. 933–945.
- [40] Summers, J. D., and Shah, J. J., 2010, "Mechanical Engineering Design Complexity Metrics: Size, Coupling, and Solvability," *ASME J. Mech. Des.*, **132**(2), p. 021004.
- [41] Lloyd, S., 2001, "Measures of Complexity: A Nonexhaustive List," *IEEE Control Syst. Mag.*, **21**(4), pp. 7–8.
- [42] Sheard, S., Cook, S., Honour, E., Hybertson, D., Krupa, J., McEver, J., and White, B., 2015, *A Complexity Primer for Systems Engineers*, INCOSE Publications, San Diego, CA, pp. 1–17.
- [43] Moses, J., 2004, "Foundational Issues in Engineering Systems: A Framing Paper," *Engineering Systems Symposium*, Cambridge, MA, Mar. 29–31.
- [44] Hennig, A., Topcu, T. G., and Szajnar, Z., 2022, "So You Think Your System Is Complex?: Why and How Existing Complexity Measures Rarely Agree," *ASME J. Mech. Des.*, **144**(4), p. 041401.
- [45] Lindemann, U., Maurer, M., and Braun, T., 2008, *Structural Complexity Management: An Approach for the Field of Product Design*, Springer Science & Business Media, New York.
- [46] Kossiakoff, A., Sweet, W. N., Seymour, S. J., and Biemer, S. M., 2011, *Systems Engineering Principles and Practice*, Vol. 83, John Wiley & Sons, Hoboken, NJ.
- [47] De Weck, O. L., Roos, D., and Magee, C. L., 2011, *Engineering Systems: Meeting Human Needs in a Complex Technological World*, MIT Press, Cambridge, MA.
- [48] Baldwin, C. Y., Clark, K. B., and Clark, K. B., 2000, *Design Rules: The Power of Modularity*, Vol. 1, MIT Press, Cambridge, MA.
- [49] Suh, N. P., 2005, "Complexity in Engineering," *CIRP Ann.*, **54**(2), pp. 46–63.
- [50] Rechten, E., and Maier, M. W., 2010, *The Art of Systems Architecting*, CRC Press, Boca Raton, FL.
- [51] Cameron, B., Crawley, E., and Selva, D., 2016, *Systems Architecture. Strategy and Product Development for Complex Systems*, Pearson Education, Upper Saddle River, NJ.
- [52] Min, G., Suh, E. S., and Hölttä-Otto, K., 2016, "System Architecture, Level of Decomposition, and Structural Complexity: Analysis and Observations," *ASME J. Mech. Des.*, **138**(2), p. 021102.
- [53] McCabe, T. J., 1976, "A Complexity Measure," *IEEE Trans. Software Eng.*, **SE2**(4), pp. 308–320.
- [54] Halstead, M. H., 1977, *Elements of Software Science (Operating and Programming Systems Series)*, Elsevier Science Inc.
- [55] Hölttä, K. M., and Otto, K. N., 2005, "Incorporating Design Effort Complexity Measures in Product Architectural Design and Assessment," *Des. Stud.*, **26**(5), pp. 463–485.
- [56] Sinha, K., and de Weck, O. L., 2016, "Empirical Validation of Structural Complexity Metric and Complexity Management for Engineering Systems," *Syst. Eng.*, **19**(3), pp. 193–206.
- [57] Sinha, K., 2014, "Structural Complexity and Its Implications for Design of Cyber-Physical Systems," *Ph.D. Dissertation*, MIT, Cambridge, MA.
- [58] Broniatowski, D. A., and Moses, J., 2016, "Measuring Flexibility, Descriptive Complexity, and Rework Potential in Generic System Architectures," *Syst. Eng.*, **19**(3), pp. 207–221.
- [59] Prokopenko, M., Boschetti, F., and Ryan, A. J., 2009, "An Information-Theoretic Primer on Complexity, Self-Organization, and Emergence," *Complexity*, **15**(1), pp. 11–28.
- [60] Ashby, W. R., 1961, *An Introduction to Cybernetics*, Chapman & Hall Ltd.
- [61] Jaynes, E. T., 1957, "Information Theory and Statistical Mechanics," *Phys. Rev.*, **106**(4), pp. 620–630.
- [62] Humann, J., Khani, N., and Jin, Y., 2016, "Adaptability Tradeoffs in the Design of Self-Organizing Systems," *Intl. Design Engineering Technical Conferences*, Charlotte, NC, Aug. 21–24.
- [63] Jones, C., and Mataric, M. J., 2003, "Adaptive Division of Labor in Large-Scale Minimalist Multi-Robot Systems," *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, Oct. 27–31.
- [64] Groß, R., Bonani, M., Mondada, F., and Dorigo, M., 2006, "Autonomous Self-Assembly in Swarm-Bots," *IEEE Trans. Rob.*, **22**(6), pp. 1115–1130.