Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343

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[1]
Department of Aerospace and Mechanical Engineering
University of Southern California
3650 McClintock Avenue, OHE 430, Los Angeles, CA, 90089
e-mail: yjin@usc.edu

ABSTRACT

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 paper, 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.

Keywords: *complex systems, self-organizing system, entropy, multiagent deep reinforcement learning*

[1] Corresponding author.

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343

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?

In the rest of this paper, Section 2 reviews the relevant work in artificial self-organizing systems, multiagent RL, and complexity theory. In Section 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 Section 5, and the conclusions drawn in Section 6, together with future work directions.

2 RELATED WORK

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 above 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 non-stationary 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 literature. Tampuu [17] extended the previous Q-learning to deep neural networks and applied a DQN [20] to train two independent agents playing the game Pong [17]. Foerster 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, he analyzed his replay stabilization methods for independent Q-learning based on StarCraft combat scenarios [22].

As a multiagent environment is usually partially observable, Hauskneche & Stone [23] used deep recurrent networks such as LSTM [24] or GRU [25] to speed up learning when agents are learning over long periods. Lowe et al. developed Multiagent Deep Deterministic Policy

Gradient (MADDPG) 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 [60-65]. 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 non-simple ways [31], researchers from different fields have sought to measure system complexity [32-34]. More than 50 complexity measures have been proposed in the literature on Engineering Design and System Engineering literature [33,35-37]. 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 [38]. System complexity grows as the system size increases because larger systems can generate more unique system states [31, 39-41]. It is also agreed that increasing the interconnections within a system increases its complexity [31,39-41]. Structuring or minimizing random interconnections between system components, such as introducing a modular structure, reduces the system's complexity [42-45].

Considerable research has been conducted on measuring complexity. Min et al. analyzed the structural complexity of complex engineering systems at various decomposition levels [46]. McCabe's cyclomatic complexity (MCC) is the first complexity measurement that utilizes the graph theory conceptualization [47]. 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 [48]. Hölttä-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 [49]. 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 [50]. 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 modulization [51,52]. McComb et al. [60]

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 modulization so that the system can be understood with less cognitive effort [53].

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 paper.

3 TASK CONSTRAINTS AND EFFECTIVE ENTROPY

This work started from two basic ideas. First, we assume AsAshby'saw 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 greater potential to complete the task. Information entropy is a common practice for measuring various states [54]. 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 **Equation (1)**:

$$Information\ Entropy = -\sum_{i}^{N} p_i \log p_i \tag{1}$$

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 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, **Equation (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 Equation (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 10-agent team could move the box much faster than the 5-agent team. Similarly, if we use these two teams to rotate the

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343
Copyright © 2024 by ASME

box, the 10-agent team can exert a maximum torque double that of the 5-agent team; therefore, within a unit of time, the 10-agent team will be able to turn the box with a much larger maximum degree compared to the 5-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:

$$A = A^e \cup A^i \tag{2}$$

where, A^e: Team actions that are effective given the task constraints,

A^i: 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 5 robots, then the other 5 robots in a 10-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.

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343

For instance, if a 10-agent team attempts to rotate a box, a single team action might achieve a maximum of 50 degrees rotation, assuming each agent contributes to a 5-degree rotation. Yet, with task constraints C_T encompassing *Narrow Pathways* and *Obstacles*, any team action resulting in more than a 20-degree rotation becomes ineffective, as it would cause collisions with obstacles or pathway borders. In this context, the 20-degree rotation defines the *team action threshold*.

The concept of the team action threshold plays a significant role in multiagent reinforcement learning (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

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343
Copyright © 2024 by ASME

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 $A^e = \{a_1^e, a_2^e, \cdots, a_M^e\}$, we have:

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

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

As mentioned above, 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 *SimRuns* = *Episodes* x *RunsPerEpisode* for training to obtain a successful learning team of that size. If there are different numbers of team sizes, *TeamSizeNum*, then the total simulation runs needed will be

$$TotalSimRuns = TeamSizeNum \times (Episodes \times RunsPerEpisode) \tag{4}$$

Consider a team with 10 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 x 100 x 40,000 = 40,000,000. If one *OneEpisodeRun* takes 0.5 seconds on a powerful PC, the *total simulation time* can be 33 weeks. In this paper, 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 **Equation (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 $E_t = \{r_1, r_2, \cdots, r_R\}$, i.e., the task environment is composed of different regions (or aspects), $r_1, r_2, \cdots$. 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 abovementioned 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

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343
Copyright © 2024 by ASME

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 above, a workflow for computationally predicting the optimal team size of SOS is shown in **Figure 1**.

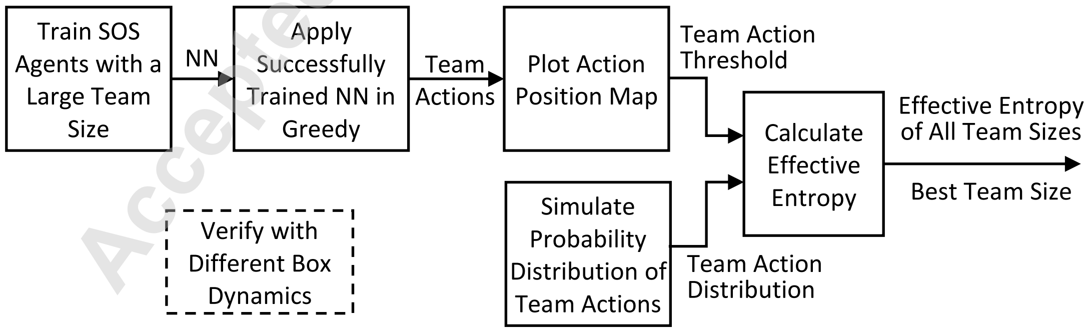

Train SOS Agents with a Large Team Size
NN
Apply Successfully Trained NN in Greedy
Team Actions
Plot Action Position Map
Team Action Threshold
Calculate Effective Entropy
Effective Entropy of All Team Sizes
Best Team Size
Simulate Probability Distribution of Team Actions
Team Action Distribution
Verify with Different Box Dynamics

Figure 1: A computing workflow of optimal team size prediction of SOS

The first step is to train an agent team of a large size (e.g., 10 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. Based on 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 simulation runs for training are reduced to:

$$TotalSimRuns = 1 \times (RunsPerEpisode \times Episodes) \tag{5}$$

which is only a fraction (1/TeamSizeNum) of the original simulation runs in **Equation (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 **Figure 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, **Figure 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 1-unit box movement along the negative x-direction**,** a 0-unit movement along the y-direction**,** and a 1-unit rotation. Such box

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343

movement can be expressed with a vector: $< x, y, \varphi > = < -1, 0, 1 >$. Similarly, we can get the movement vector of the box when an agent is in different regions of the box neighborhood:

- Region 1: $< -1, 0, 1 >$, Region 2: $< -1, 0, -1 >$, Region 3: $< 1, 0, 1 >$
- Region 4: $< 1, 0, -1 >$, Region 5: $< 0, -1, 0 >$, Region 6: $< 0, 1, 0 >$

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

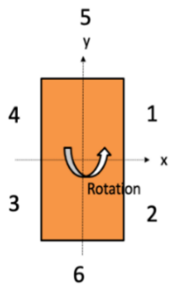
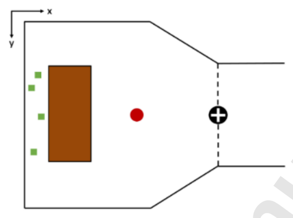
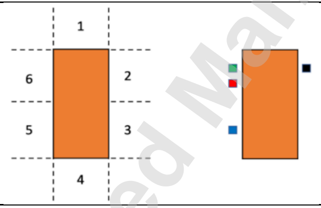
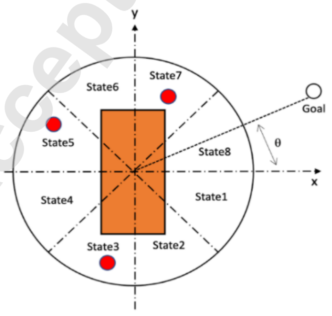


Figure 2: Box-pushing task state representation with numbers indicating box regions.

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

Parameters	Illustration or Equation	Description
The box-pushing problem		**Task**: The solid rectangular box is supposed to be moved into the goal position marked with"+" **Agents**: The green squares have limited sensing and pushing capabilities to push and rotate the box. **Constraints**: The center dot obstacle and the side walls. The box cannot hit the obstacle or side walls.
Box neighborhood		**Regions** and **Step**: 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		**State space**: The task state space is defined by dividing the box's vicinity into 8 segments. In addition, the box's orientation angle β 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° for easier training. **Action space**: 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	$R_{total} = w_1 R_{dis} + w_2 R_{rot} + w_3 R_{col} + w_4 R_{goal}$	**Reward**: The *total reward* is a weighted summation of

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343

		the reward items: *distance*, *rotation*, *collision*, and *goal*. All weights have fixed values (see [8]).
Single-agent learning	Q-Learning: $Q_{t+1}(s,a) = E\left[r+\gamma \max_a Q_i(s',a')\vert s,a\right]$ Deep Q-learning Network (DQN): Loss function: $L_i(\theta_i) = E\left[(r+\gamma \max_a Q_i(s',a';\theta_{i-1}) - Q_t(s,a;\theta_i))^2\right]$	**Deep Q-learning:** A deep Q-learning network (DQN) is used to approximate Q-table in Q-learning. *Value calculation:* Each agent *i* has its own *Q-value network* = $Q_t(\theta_i)$, which is updated through training by minimizing MSE loss $L_i(\theta_i)$. *Action selection:* Each agent *i* takes an ε-greedy policy to select its next action.
Multiagent learning	Common/shared information: $S = \{s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8,s_9\}$ $R_{total} = w_1R_{dis} + w_2R_{rot} + w_3R_{col} + w_4R_{goal}$ Individual/private information: *Q-value network* = $Q_t(\theta_i)$,	**Multiagent deep Q-learning**: (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.

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.

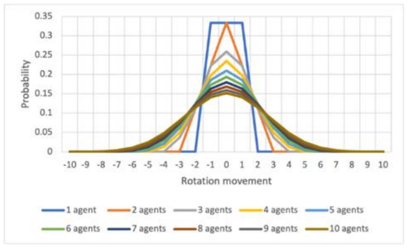
4.2 Effective entropy and optimal size prediction – A case study

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 degrees, meaning each agent can contribute 5 (-5) degrees of box rotation if positioned in regions 1 or 3 (2 or 4) and 0 degrees 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 **Figure 3**(a).

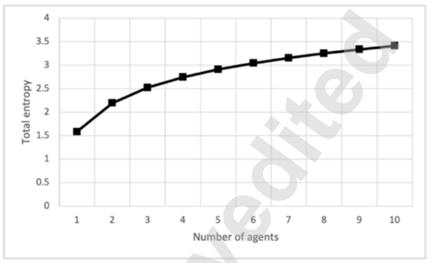
Obstacle constraints: Based on **Equation (1)**, the resulting total entropy of each agent team size is shown in **Figure 3**(b), which does not consider constraints; the total entropy of the

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343

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.



(a) Probability distribution of rotation vs. agent team size



(b) Total entropy vs. agent team size

Figure 3: Multiagent box-pushing task without considering constraints

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.

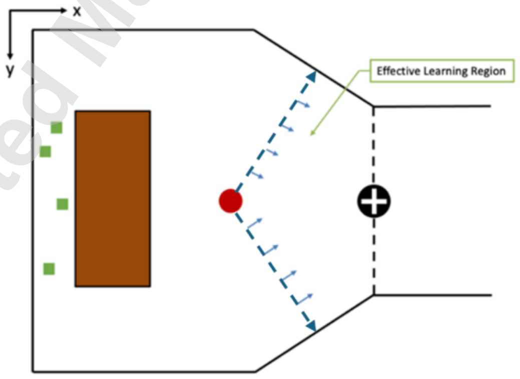


Figure 4: Effective learning region of Box-pushing task

Rotation maps: Following the workflow in **Figure 1**, we first train the TeamSize = 10 team with 100 random seeds [8], then we apply the successfully trained neural networks *in greedy* and

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343

plot *rotation maps* of the box, as shown in **Figure 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-degree rotation **Figure 5** (5 deg), the box is pushed to rotate 5 degrees 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 degrees, they rarely happen and hence are missing for box-pushing. They, thus, should be excluded from the effective entropy calculation. Therefore, 15 degrees of rotation is the ***rotation threshold*** for *effective entropy* calculation.

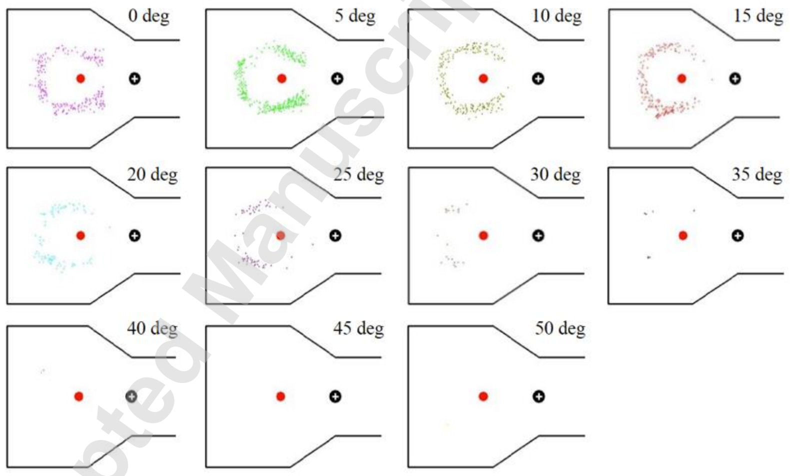


Figure 5: Action maps of different rotation movements with 10 agents

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

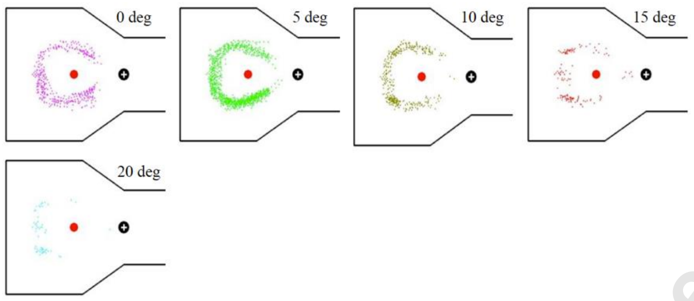


Figure 6: Action maps of different rotation movements with 4 agents

It is worth emphasizing that although the team action maps, as illustrated in **Figures 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 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 degrees, 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 degrees of rotation. Each state's corresponding probability distribution is simulated using Monte Carlo simulation (see **Figure 3**(a). For a small number of agents, e.g., 2 agents, the possible rotation states are -10, -5, 0, 5, and 10 degrees, 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 degrees. However, only 7 states between -15 to 15 degrees are effective. Thus, the effective entropy should only calculate

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343

the probability distributions up to these 7 rotation states.

The plots of the effective entropy for different team sizes are shown in **Figure 7**, which indicates that five (5) 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 **Figure 8**. The performance score in **Figure 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 [58].

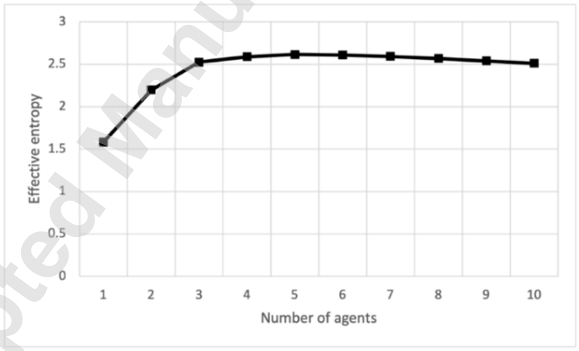


Figure 7: Effective entropy with varying numbers of agents in box-pushing task

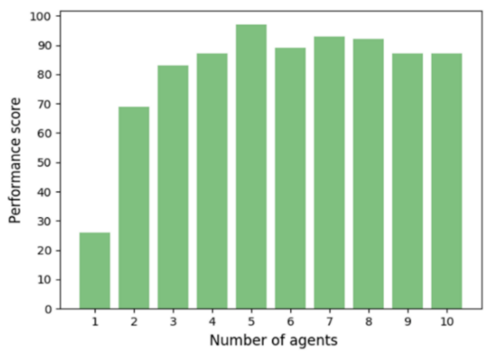


Figure 8: Performance score with different numbers of agents in box-pushing task [8]

4.3 Verification

In order 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 degrees, instead of 5 degrees, 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 **Figure 9**, which indicates that 10 degrees of rotation should be the rotation threshold, as box centers rarely pass the effective learning region beyond this degree. Note that 10 degrees of rotation are slightly smaller than the 15 degrees 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.

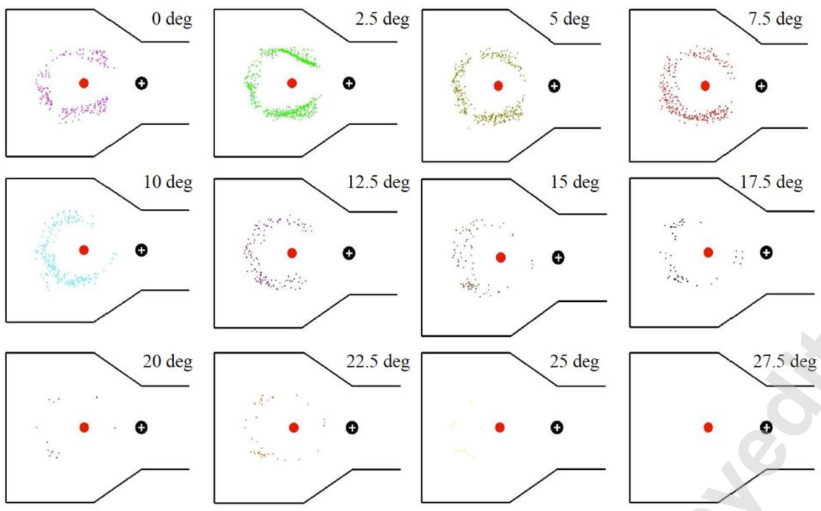


Figure 9: Action maps of different rotation movements with 20 agents

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 calculate the effective entropy with different team sizes, as shown in **Figure 10**. The effective entropy first increases up to a point and then declines, with 8 and 9 agents obtaining the maximum effective entropy.

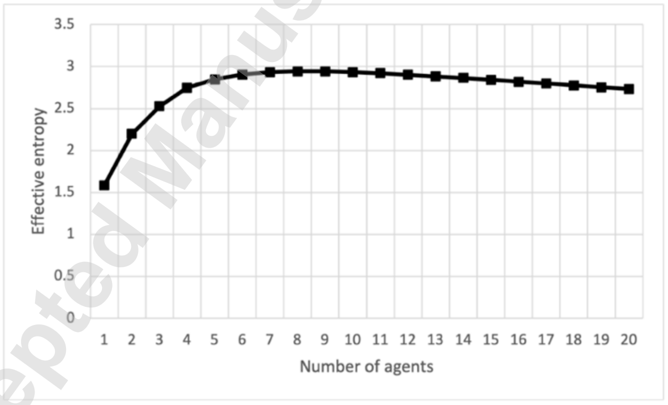


Figure 10: Effective entropy with varying numbers of agents in 20-agent box-pushing

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

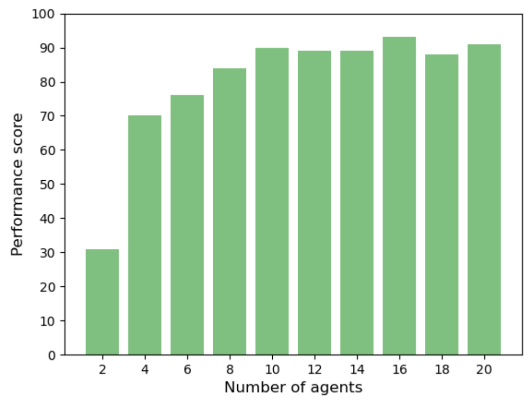


Figure 11: Performance score with different numbers of agents in heaving box-pushing

The performance scores are quite different across team sizes in the previous case (**Figure 8**). As depicted in **Figure 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 Section 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 5 agents to 10 agents. Although, in this case, the effect is straightforward, the outcomes affirm that the proposed effective entropy measure and the method described above 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 above is based on two principles. First, Ashby's law of requisite 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 reinforcement learning (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 [59], 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 paper 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, 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

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343

as "*task constraints*" that influence the execution and effect of the pivotal team actions. The obstacles and walls in **Figure 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 in **Figure 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.

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 paper can provide valuable insights for dealing with such problems. McComb et al. [60] 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 Figure 4b of [60] indicates, 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.”

Journal of Computing and Information Science in Engineering. Received February 13, 2023;
Accepted manuscript posted March 23, 2024. doi:10.1115/1.4065343

In a separate study, Hulse et al. [61] applied an adapted multiagent reinforcement learning method as an optimization algorithm to solving a quadrocopter 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 paper 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 above.

- 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. Additionally, 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. Additionally, 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.

7 ACKNOWLEDGMENT

This paper 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.

8 REFERENCES

[1] Reynolds, C. W. (1987, August). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (pp. 25-34).

[2] Ashby, W. R. (1991). Requisite variety and its implications for the control of complex systems. In *Facets of systems science* (pp. 405-417). Springer, Boston, MA.

[3] Chiang, W., & Jin, Y. (2012, August). Design of Cellular Self-Organizing Systems. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Vol. 45028, pp. 511-521). American Society of Mechanical Engineers.

[4] Humann, J., Khani, N., & Jin, Y. (2014). Evolutionary computational synthesis of self-organizing systems. *AI EDAM*, *28*(3), 259-275.

[5] Khani, N., Humann, J., & Jin, Y. (2016). Effect of social structuring in self-organizing systems. *Journal of Mechanical Design*, *138*(4).

[6] Khani, N., & Jin, Y. (2015). Dynamic structuring in cellular self-organizing systems. In *Design Computing aCognition'14'14* (pp. 3-20). Springer, Cham.

[7] Ji, H., & Jin, Y. (2018). Modeling Trust in Self-Organizing Systems With Heterogeneity. In *ASME 2018 International Design Engineering Technical Conferences and Computers and*

Information in Engineering Conference. American Society of Mechanical Engineers Digital Collection.

[8] Ji, H., & Jin, Y. (2021). Evaluating the learning and performance characteristics of self-organizing systems with different task features. AI EDAM, 35(4), 404-422.

[9] Ji, H., & Jin, Y. (2022). Knowledge Acquisition of Self-Organizing Systems With Deep Multiagent Reinforcement Learning. *Journal of Computing and Information Science in Engineering*, *22*(2).

[10] Dasgupta, P. (2008). A multiagent swarming system for distributed automatic target recognition using unmanned aerial vehicles. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *38*(3), 549-563.

[11] Ruini, F., & Cangelosi, A. (2009). Extending the Evolutionary Robotics approach to flying machines: An application to MAV teams. *Neural Networks*, *22*(5-6), 812-821.

[12] Lamont, G. B., Slear, J. N., & Melendez, K. (2007, April). UAV swarm mission planning and routing using multi-objective evolutionary algorithms. In *2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making* (pp. 10-20). IEEE.

[13] Wei, Y., Madey, G. R., & Blake, M. B. (2013, April). Agent-based simulation for uav swarm mission planning and execution. In *Proceedings of the Agent-Directed Simulation Symposium* (pp. 1-8).

[14] Chen, C., & Jin, Y. (2011, January). A behavior based approach to cellular self-organizing systems design. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Vol. 54860, pp. 95-107).

[15] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[16] Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(2), 156-172.

[17] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., ... & Vicente, R. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, *12*(4), e0172395.

[18] Tan, M. (1993). Multiagent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning* (pp. 330-337).

[19] Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Ph.D. Dissertation, Cambridge University, Cambridge, England.

[20] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529-533.

[21] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018, April). Counterfactual multiagent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).

[22] Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., & Whiteson, S. (2017, July). Stabilising experience replay for deep multiagent reinforcement learning. In *International conference on machine learning* (pp. 1146-1155). PMLR.

[23] Hausknecht, M., & Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*.

[24] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[25] Chung, J., Gulcehre, C., Cho, K., & 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., & Mordatch, I. (2017). Multiagent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*.

[27] Brown, N., & Sandholm, T. (2019). Superhuman AI for multiplayer poker. *Science*, *365*(6456), 885-890.

[28] Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., & Mordatch, I. (2019). Emergent tool use from multiagent autocurricula. *arXiv preprint arXiv:1909.07528*.

[29] Wu, S. A., Wang, R. E., Evans, J. A., Tenenbaum, J. B., Parkes, D. C., & Kleiman-Weiner, M. (2021). Too Many Cooks: Bayesian Inference for Coordinating Multiagent Collaboration. *Topics in Cognitive Science*, *13*(2), 414-432.

[30] Bar-Yam, Y. (2002). General features of complex systems. *Encyclopedia of Life Support Systems (EOLSS), UNESCO, EOLSS Publishers, Oxford, UK*, *1*.

[31] Simon, H. A. (1991). The architecture of complexity. In *Facets of systems science* (pp. 457-476). Springer, Boston, MA.

[32] Bashir, H. A., & Thomson, V. (1999). Estimating design complexity. *Journal of Engineering Design*, *10*(3), 247-257.

[33] Sheard, S. A., & Mostashari, A. (2010, July). 7.3. 1 A complexity typology for systems engineering. In *INCOSE International Symposium* (Vol. 20, No. 1, pp. 933-945).

[34] Summers, J. D., & Shah, J. J. (2010). Mechanical engineering design complexity metrics: size, coupling, and solvability. *Journal of Mechanical Design*, *132*(2).

[35] Lloyd, S. (2001). Measures of complexity: a nonexhaustive list. *IEEE Control Systems Magazine*, *21*(4), 7-8.

[36] Sheard, S., Cook, S., Honour, E., Hybertson, D., Krupa, J., McEver, J., ... & White, B. (2015). A complexity primer for systems engineers. *INCOSE Complex Systems Working Group White Paper*, *1*(1), 1-10.

[37] Moses, J. (2004). Foundational issues in engineering systems: A framing paper. *Engineering Systems Monograph*, *2*, 1-15.

[38] Hennig, A., Topcu, T. G., & Szajnfarber, Z. (2022). So You Think Your System Is Complex?: Why and How Existing Complexity Measures Rarely Agree. *Journal of Mechanical Design*, *144*(4).

[39] Lindemann, U., Maurer, M., & Braun, T. (2008). Structural complexity management: an approach for the field of product design. Springer Science & Business Media.

[40] Kossiakoff, A., Sweet, W. N., Seymour, S. J., & Biemer, S. M. (2011). Systems engineering principles and practice (Vol. 83). John Wiley & Sons.

[41] De Weck, O. L., Roos, D., & Magee, C. L. (2011). Engineering systems: Meeting human needs in a complex technological world. Mit Press.

[42] Baldwin, C. Y., Clark, K. B., & Clark, K. B. (2000). Design rules: The power of modularity (Vol. 1). MIT press.

[43] Suh, N. P. (2005). Complexity in engineering. CIRP annals, 54(2), 46-63.

[44] Rechtin, E., & Maier, M. W. (2010). The art of systems architecting. CRC press.

[45] Cameron, B., Crawley, E., & Selva, D. (2016). Systems Architecture. Strategy and product development for complex systems. Pearson Education.

[46] Min, G., Suh, E. S., & Hölttä-Otto, K. (2016). System architecture, level of decomposition, and structural complexity: analysis and observations. *Journal of Mechanical Design*, *138*(2).

[47] McCabe, T. J. (1976). A complexity measure. IEEE Transactions on software Engineering, (4), 308-320.

[48] Halstead, M. H. (1977). Elements of Software Science (Operating and programming systems series). Elsevier Science Inc.

[49] Hölttä, K. M., & Otto, K. N. (2005). Incorporating design effort complexity measures in product architectural design and assessment. Design studies, 26(5), 463-485.

[50] Summers, J. D., & Shah, J. J. (2010). Mechanical engineering design complexity metrics: size, coupling, and solvability. Journal of Mechanical Design, 132(2).

[51] Sinha, K., & de Weck, O. L. (2016). Empirical validation of structural complexity metric and complexity management for engineering systems. Systems Engineering, 19(3), 193-206.

[52] Sinha, K. (2014). Structural complexity and its implications for design of cyber-physical systems (Doctoral dissertation, Massachusetts Institute of Technology).

[53] Broniatowski, D. A., & Moses, J. (2016). Measuring flexibility, descriptive complexity, and rework potential in generic system architectures. Systems Engineering, 19(3), 207-221.

[54] Prokopenko, M., Boschetti, F., & Ryan, A. J. (2009). An information-theoretic primer on complexity, self-organization, and emergence. Complexity, 15(1), 11-28.

[55] Humann, J., Khani, N., & Jin, Y. (2016, August). Adaptability tradeoffs in the design of self-organizing systems. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 50190, p. V007T06A016). American Society of Mechanical Engineers.

[56] Jones, C., & Mataric, M. J. (2003, October). Adaptive division of labor in large-scale minimalist multi-robot systems. In Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No. 03CH37453) (Vol. 2, pp. 1969-1974). IEEE.

[57] Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. IEEE transactions on robotics, 22(6), 1115-1130.

[58] Ashby, W. R. (1961). *An introduction to cybernetics*. Chapman & Hall Ltd.

[59] Jaynes, E. T. (1957) Information Theory and Statistical Mechanics, Phys. Rev. 106, 620; doi:10.1103/PhysRev.106.620

[60] McComb, C. Cagan, J. Kotovsky, K. (2017) Optimizing Design Teams Based on Problem Properties: Computational Team Simulations and an Applied Empirical Test, *J. Mech. Des*. Apr 2017, 139(4): 041101 (12 pages)

[61] Hulse D, Tumer K, Hoyle C, Tumer I (2019). Modeling multidisciplinary design with multiagent learning. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 33, 85–99.

[62] Chen, L. Li, S. (2005) Analysis of Decomposability and Complexity for Design Problems in the Context of Decomposition, *J. Mech. Des.* Jul 2005, 127(4): 545-557

[63] Joshua D. Summers, J.D. and Shah, J. (2010) Mechanical Engineering Design Complexity Metrics: Size, Coupling, and Solvability *J. Mech. Des.* Feb 2010, 132(2): 021004 (11 pages)

[64] Allaire, D., He, Q., Deyst, J., Willcox, K. (2012) An Information-Theoretic Metric of System Complexity With Application to Engineering System Design *J. Mech. Des.* October 2012, 134(10): 100906.

[65] Chen, W., Fuge, M., Chazan, J. Design Manifolds Capture the Intrinsic Complexity and Dimension of Design Spaces, *J. Mech. Des.* May 2017, 139(5): 051102.