Check for updates

Proceedings of the ASME 2025
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC-CIE2025
August 17-20, 2025, Anaheim, CA

DETC2025-169584

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

EMERGENT ROLE SPECIALIZATION IN SELF-ORGANIZED SYSTEMS VIA K-MEANS-BASED BEHAVIORAL ANALYSIS

Bingling Huang*
Dept. of Mechanical Engineering
California State University, Fullerton.
Fullerton, USA
bihuang@fullerton.edu
(*Corresponding Author)

Maximillian Haaga
Dept. of Mechanical Engineering
California State University, Fullerton
Fullerton, USA
mhaaga@csu.fullerton.edu

Hao Ji
Center for Advanced Research Computing
University of Southern California
Los Angeles, USA
haoji@usc.edu

ABSTRACT

Self-organized system (SOS) offers a compelling paradigm for enabling autonomous coordination in a multiagent system (MAS). By leveraging decentralized decision-making, these systems can dynamically adapt to evolving environments, making them highly suitable for complex engineering applications. While multiagent reinforcement learning (MARL) has advanced agents to perform collaboratively, the mechanisms driving such self-organization still remain largely unexplored. Thus, this paper aims to deepen the understanding how agents self-organize themselves into an intelligent team from the role specialization perspective. In the context of a precision assembly task with collision avoidance, agent teams are trained to collaborate without predefined role assignments. We apply physical motion analysis to quantify individual contributions to system dynamics as a team and utilize K-means clustering to further provide a data-driven view of emergent behaviors. By analyzing agent role's scores based on their contributions to the system, the mechanisms of role specialization have been uncovered. Results show that the effective role differentiation can be naturally emergent from the MARL training process. Furthermore, the training leads the role assignments into a hierarchical structure, where some agents take primary roles while others provide dynamic support. The findings offer practical guidelines for designing an adaptive, efficient, and robust multiagent systems for applications such as autonomous robotics and advanced manufacturing.

Keywords: Self-organized systems, multiagent reinforcement learning, role specialization, K-means clustering, system design.

1. INTRODUCTION

Robotic collaboration has become a widely adopted approach to achieving high levels of autonomy in engineering applications [1]. From the industrial manufacturing line [2] to hazardous environments in rescue missions [3], multiagent systems (MAS) offer promising solutions for tackling complex tasks. However, traditional rule-based systems often struggle to adapt to dynamic environments as they rely on predefined instructions [4]. This limitation has led to bring our interests to self-organizing systems (SOSs), where agents spontaneously form complex group behaviors through localized observation and interactions in response to changing external circumstances [4]. Such systems exhibit a high degree of adaptability [5], making them well-suited for environments that demand autonomy. Additionally, SOSs reduce the need for a top-down system design approach [6], which becomes increasingly challenging as tasks complexity grows [4].

Learning-based approaches, particularly multiagent reinforcement learning (MARL) [7], have been widely applied to aid the design of SOSs. MARL enables agents to independently learn optimizing performance through interacting with environments. This results in the emergence of adaptive behaviors that are not explicitly preprogrammed. However, the training process occurs in an end-to-end black-box manner, making it difficult to interpret how emergent group behaviors form. Such lack of understanding introduces unpredictability into the system design, posing challenges in controlling system outcomes.

To address this issue, this paper focuses on delving into the emergent role division within the MARL training process. Specifically, we seek to explain the underlying mechanisms that drive the self-organization. Understanding these emergent patterns can provide valuable insights into the optimal structuring of agent

Copyright © 2025 by ASME

teams, including determining effective team sizes, predefining roles, and optimizing system design for specific engineering applications.

To achieve this, we conduct experiments in a precision assembly task involving collision avoidance, where agents must collaboratively transport and assemble objects while avoiding obstacles. This task requires coordination and adaptive decision-making for a team. Based on the training results, we conduct role division analysis through both physical and K-means clustering approaches. The physical analysis provides an explicit assessment of each agent's contribution to system dynamics and motion, serving as a foundation for understanding role differentiation. Building upon this, the data-driven analysis offers a broader, system-level perspective, capturing behavioral patterns and emergent teamwork structures directly from training data beyond the physical impact. The studies aim to answer research questions *1) Do emergent roles naturally form in SOS training? 2) If so, what behavioral patterns and role structures contribute to efficient coordination?* It is hoped to uncover the mechanism that drives more effective and adaptive self-organized teams.

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 illustrates the employed methodologies, experimental design and training settings. Section 4 presents the training and analysis results. Section 5 offers a thorough discussion of key findings. Finally, Section 6 concludes the study and outlines future directions.

2. RELATED WORK

MARL has grown in recent years by leaps and bounds. This method trains a group of agents with optimal policies through maximizing their reward through interaction with the environment [8]. During the training process, the end-to-end nature of MARL often leads to black-box behavior, making it difficult for humans to interpret the underlying mechanisms. One of the key challenges is understanding how role specialization emerges, an insight that is essential for interpreting, predicting, and ultimately controlling system behavior. Role specialization has been explored in domains such as robotics and swarm intelligence.

Some works impose predefined roles [9], while others explore emergent roles in swarm systems via simple local rules [10]. However, these methods are often insufficient for addressing the complex demands of modern engineering tasks.

More recent efforts have explored unsupervised role discovery in MARL. For instance, the RODE learning framework [11] and the ROMA framework [12] are proposed for the investigation of natural role division in MAS. In [11] and [12], the authors presents a framework of MARL that included roles within their learning models, where roles are defined as the specific duties and or tasks that certain agents take on. The RODE framework's model broke down physical tasks into different sub tasks, each of which being associated with a role. This method was able to simplify the division of roles into action spaces, which led to higher efficiency in learning and assigning roles to agents. The ROMA framework trains the agents in different roles, and these roles will influence their policies and rewards. The role encoder is built within the agent's learning framework, with each training episode it learns to diversify roles better based on the agent's positioning. The ROMA framework analyzed the division of roles into three learned role embedding maps. This paper was able to make a connection between the division of roles and an increased success rate for the team of homogenous agents. The RODE framework took a different approach by plotting the actions of each agent onto a graph. Thus, the role division was interpreted through the decomposition of joint action spaces through individual action effects.

Chen et al. have explored a similar topic, focusing on the emergent behaviors of agents in different roles within a manufacturing task [13]. This paper assigned agents different tasks to see how well they would work together. This is an example of a heterogenous set of agents that each have access to different rewards and policies based on their use. A problem that arises in homogeneous systems is the ability to interpret the division of these roles, which is important for addressing the issue of partial observability in MARL systems.

In [14], the authors differentiate roles by dividing reward functions into differing roles. This established that different roles directly correlated to a more effective outcome. Another study proposed categorizing the roles of Large Language Model (LLM) based agent systems [15]. These were determined through pre-defined, model generated, and data derived systems of categorizing an agent's profile. Another study proposed that the use of k-means clustering techniques for finding agent categorization was possible [16]. However, this paper utilized the Agent Categorization (AC) problem as a mathematical function, not simulation. Thus, k-means clustering techniques could be a valid way to analyze the division of roles in our study. A methodology utilized this leader-follower relationship, wherein the "leader", in this case called the expert, had prior knowledge over the one follower [17]. This idea of differing roles was further explored by other papers, seeking to determine what would happen in systems with more than just two homogenous agents.

However, the absence of clear role differentiation in engineering applications can result in inefficient coordination, and even conflicting actions among agents. Those will ultimately increase operational costs and reduce the overall system performance. In contrast, effective role division allows agents to specialize in complementary functions, enhancing both efficiency and collaborative success. Motivated by this, we will investigate the underlying mechanisms of role specialization in learning-based SOSs.

3. METHODS

3.1 Multi-agent reinforcement learning and algorithm

Multi-agent reinforcement learning (MARL) has been widely applied to solve tasks involving multiple agents, typically modeled as a Markov Decision Process (MDP) [18]. In this study, we focus on cooperative agents that interact with and influence their environment in the context of complex engineering tasks. The MARL framework is formulated as a tuple of $< S, A, P, R, \gamma, N, t >$ in a sequence of discrete-time steps [19].

In the model, $t \in \mathbf{R}$ denotes the time step; N represents the number of agents; $\mathbf{S}$ means environmental state information; $\mathbf{A}$ defines the joint actions space, where $a_t^i \in \mathbf{A}$ corresponds to the

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

action chosen by agent i at time step t. The joint action set is defined as $A = A^1 \times \cdots \times A^N$ [7]. The transition function, $P(s_{t+1}|s_t, a_t): S \times A \times S \rightarrow [0, 1]$, governs state transitions. The joint policy $^{S}\boldsymbol{\Pi}$ is represented as $^{S}\boldsymbol{\Pi} \sim {}^{S}\boldsymbol{\Pi}^{1} \times {}^{S}\boldsymbol{\Pi}^{2} \times \ldots \times {}^{S}\boldsymbol{\Pi}^{N}$, indicating the learned policies of all agents. The reward function, $\mathbf{R}$, provides a shared team reward to encourage cooperative behavior in pursuit of a common goal in the context of collaborative tasks. The reward functions use numerical scores to assess the team performance. Additionally, the discount factor, γ, determines how future rewards influence the present value of decision. The goal of solving an MDP is to determine an optimal policy π that can maximize the cumulative reward [20].

Traditional Q-learning, a tabular-based approach, attempts to construct a state-action map to learn an optimal policy. However, due to the high-dimensional state-action space, this scalability is limited. To address this issue, Deep Q-learning algorithm [21] is employed, leveraging deep neural networks to approximate the Q-value function efficiently.

In this study, we adopt the Centralized Training Decentralized Execution (CTDE) framework, which is well-suited for cooperative MARL tasks. Under this paradigm, each agent maintains its own neural network, perceiving its peers as part of the environment. The training is centralized, using global information, while execution remains decentralized, with each agent making independent decisions based on its local observations [22] [23]. The loss function for Deep Q-learning, used for iteratively updating the Q-network's parameters, is defined in equation (1).

$$L_j(\theta_j^i) = E\left[\left(r_{t+1} + \gamma \max_{a^i} Q(s_{t+1}, a_{t+1}^i \mid {}^{-}\theta_j^i) - Q(s_t, a_t^i \mid \theta_j^i)\right)^2\right] \quad (1)$$

where i represents the ith agent, and j represents the iteration number, while θ_j^i is the parameter of the target networks and ${}^{-}\theta_j^i$ is the parameter of the online network.

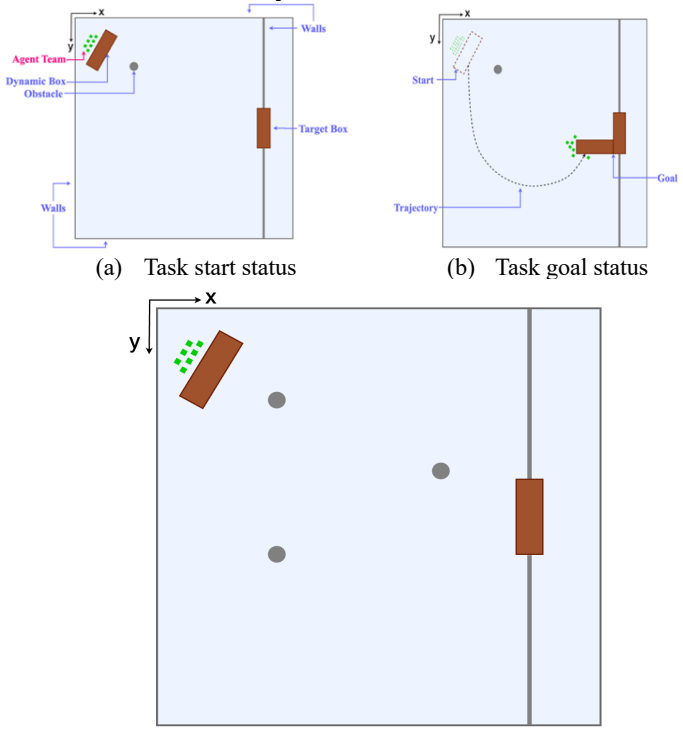
In cooperative tasks, agents are trained to maximize cumulative rewards derived from their joint actions over time, and their individual neural networks are trained concurrently to encode cooperative behaviors.

3.2 Task description

Precision assembly plays a crucial role in the engineering field [24], particularly in manufacturing processes where products are constructed by assembling individual components. Achieving high precision while ensuring collision avoidance is essential for both efficiency and safety. Consequently, our study will concentrate on an "L"-shape assembly involving collision avoidance tasks to simulate the scenarios encountered in factory settings.

Figure 1 illustrates the "L-shape" assembly tasks involving collision avoidance, where a team of agents is responsible for transporting a Dynamic Box and assembling it with a Target Box to form a "L" configuration. The initial and goal states are shown in Figure 1-(a) and Figure 1-(b), respectively.

Our study focuses on a complex assembly task characterized by three obstacles in the field, as shown in Figure 1-(c), with specifics provided in Table 1. The environment is a 1000×1000 pixels field comprising a movable Dynamic Box, a stationary Target Box, three fixed obstacles and four surrounding walls. The Dynamic Box, depicted as the upper-left brown rectangle, has a mass of 1 kg and is manipulated by the agent team, while the Target Box, positioned at the center of the right boundary. Both boxes are set size as 180×60 pixels. Agents, represented as green squares, must spontaneously self-organize to transport the Dynamic Box along an appropriate trajectory to assemble two boxes into an "L" configuration, shown as Fig.1-(b). Throughout the process, agents must ensure that the Dynamic Box does not collide with the walls or any obstacles.



(a) Task start status (b) Task goal status

(c) Task description

Figure 1: *Task environment introduction and task description*

This task's complexity is heightened due to several aspects. First of all, the box's movement physically depends on the collective impulses provided by all agents at each step. However, each agent decides only by its local information and the performance is evaluated by the whole team's common work. What's more, the presence of three stationary obstacles significantly restricts viable movement paths. It makes it challenging for agents to collaboratively transport and rotate the box to meet requirements of final configurations and avoiding collisions at the same time. Such rigorous constraints place high demands on team intelligence and coordination strategy. Therefore, achieving task success requires agents to efficiently process available information and employ an optimized system design for effective decision-making.

This paper will thus explore the emergent team performances and delve into the learning mechanism of a multiagent team in the context of "L"-assembly with collision avoidance. The tasks are simulated by *pygame* [25] environment.

Table 1: *Task settings*

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

Field size	1000×1000
Box size	180×60
Target box size	180×60
Box start center coordination	(150, 180)
Target box center coordination	(950, 500)
Obstacle center coordination	(350, 170) (350, 370) (500, 270)
Box mass (kg)	1
Push impulse ($1\ N \cdot s$)	1

3.3 Action space and state space

The agent teams are trained using Deep Q-learning [21], incorporating an epsilon-greedy (ε–$greedy$) strategy [26] to balance exploration and exploitation in the search for an optimal policy. The hyperparameters employed in training are shown in Table 2, including the number of training episodes, discount factors, learning rates, and others.

Table 2: *Hyperparameters of training*

Training episodes	20,000
Discount factor	0.99
Memory buffer size	1000
Mini-batch size	32
Target network update frequency	200
Learning rate	0.001
Neural network size	(63, 64, 128, 6)
epsilon	1 → 0.01

Action Space: In the assembly task, agent teams transport the Dynamic Box by applying pushing forces. To structure their interactions, the surrounding area of the Dynamic Box is divided into six distinct regions, as illustrated in Figure 2. During each simulation step, an agent selects one of these regions and applies a pushing force of 1 N·s to influence the box's movement. Due to the agents' small size, multiple agents can occupy the same region simultaneously, enabling coordinated force application. This arrangement facilitates collective manipulation, allowing agents to work together for precise movement and rotation of the box.

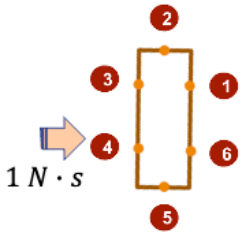


Figure 2: *Action space*

Therefore, the action space is established as follows:

$$A = < a_1, a_2, a_3, a_4, a_5, a_6 > \quad (2)$$

where a_i *means push i position with impulse of* $1\ N \cdot s$.

State Space: The state space provides environmental and surrounding information, which represented as S is set as a 63-digit tuple, shown as in equation (3). This tuple encompasses data regarding the vicinity situation around the Dynamic Box within its first 60 digits, while the last three digits details the Dynamic Box's velocity in x and y directions and its angular velocity. Thus, the *environmental state* can be defined as:

$$S = < Vicinity\ Situation, v_X, v_Y, \omega > \quad (3)$$

where *Vicinity Situation* is encoded as a tuple of binary number ("0" for absence or "1" for the presence), indicating if there exists obstacles / target box around the Dynamic Box. This is determined through sensors equipped on the Dynamic Box, which have a detection range of 200 pixels.

3.4 Reward function schema

Reward functions, including local and global rewards, are the key to guiding agents [22]. While local rewards refer to the credits based on the part of the system that an individual agent can directly observe, global rewards mean the system preference [23]. Since the assembly tasks have high task complexity [27] a shaping reward [22] is introduced in the reward function to provide additional guidance for the implementation. Therefore, the reward function schema is set as:

$$R = \omega_1 R_{local} + \omega_2 R_{global} + \omega_3 R_{shaping} \quad (4)$$

where ω_1, ω_2 and ω_3 are the hyperparameters to weigh and balance different reward terms.

They are set and tuned based on the following:

$$\omega_1 + \omega_2 + \omega_3 = 1 \quad (5)$$

In our cases, we set $\omega_1 = 0.2$, $\omega_2 = 0.5$, and $\omega_3 = 0.3$ after dirty training and tests. This setting encourages agents to pay more attention to the global and shaping rewards over the local rewards for a better consideration of the whole system rather than the local interests, which can help them succeed in their tasks.

The local rewards include a rotation reward $R_{rotation}$ to control the rotational dynamic. To prohibit rotating too many degrees in one step which may cause instability, the rotation reward is defined as:

$$R_{local} = R_{rotation} = C_r \cdot (cos(\Delta\rho_i) - \cos(\Delta\rho_{control})) \quad (6)$$

where ρ is the angle of the dynamic box's rotation. $\Delta\rho_i$ is the change of rotational angle of the box at step i. $\Delta\rho_{control}$ is a hyperparameter representing an angle change threshold for

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

receiving punishment when exceeded; it is set as 11° in all cases. C_r is a hyperparameter that works to tune the collective effect with other rewards. It is set as 10 in our cases.

The global rewards R_{global} include goal rewards R_{goal}, distance rewards $R_{distance}$ and collision rewards $R_{collision}$, formulated as below:

$$R_{global} = R_{goal} + R_{collision} + R_{distance} \quad (7)$$

The goal reward R_{goal} provides a large positive reward once the task goal is achieved and can be set as:

$$R_{goal} = C_{goal} \cdot I_{GOAL} \quad (8)$$

where I_{GOAL} is a unit indicator to check if the assembly goal with collision avoidance is achieved. C_{goal} is a hyperparameter and is set as 1000.

The collision reward $R_{collision}$ is designed to avoid any collisions that happen during the task, including Walls and Obstacles, which are set as:

$$R_{collision} = C_{collide-wall} \cdot I_{COLLIDE-wall} + C_{collide-obs} \cdot I_{COLLIDE-obs} \quad (9)$$

where $I_{COLLIDE-wall}$ is a unit indicator to check if the dynamic box collides onto the walls. $I_{COLLIDE-obs}$ is an indicator of the event to see if the dynamic box collides with the obstacles. The $C_{collide-wall}$ and $C_{collide-obs}$ are set as -100 and -200, respectively.

The distance reward $R_{distance}$ works to encourage agents to push the box toward the target box and punish them if they go in the opposite direction. It is set as:

$$R_{distance} = C_{distance} \cdot \Delta d_i \quad (10)$$

where Δd_i is the distance difference of the dynamic box that has moved to the target box at step i. $C_{distance}$ is set as 0.1 as a hyperparameter in our cases

Based on our previous research [22], the reward shaping field is applied and set as:

$$R_{shaping} = C_{shaping} \cdot \Phi(\vec{s}) = C_{shaping} \cdot h(\alpha) \cdot g(\beta) = C_{shaping} \cdot exp(-C_\alpha \cdot |\alpha - \alpha_{goal}|) \cdot exp(-C_\beta \cdot |\beta - \beta_{goal}|) \quad (11)$$

where α, β are two goal signals for shaping, which presents the angles of the dynamic box rotation and the angles between the horizontal line and the line connecting the centers of the two boxes, shown in Figure 3.

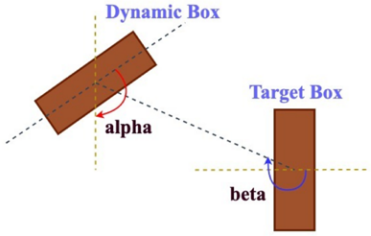


Figure 3: *Box rotation angle α and related angle β*

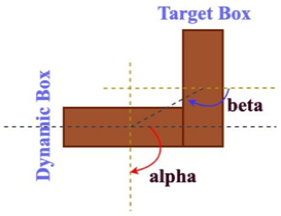


Figure 4*: Final configuration in terms of alpha and beta angles.*

It can be seen from Figure 4 that the final "L"-shape configuration can be represented by the final value of the two angles, meaning that the signal α to be 90° and the signal β to be 135° simultaneously at the end of the task. Thus, the prior information about the task goal can be set as hyperparameters to be fed into the system, and we have:

$$\alpha_{goal} = 90° \quad (12)$$
$$\beta_{goal} = 135° \quad (13)$$

As for other hyperparameters, $C_{shaping}$ works to balance the shaping rewards with others and is set as 1000 in all cases. Besides, C_α and C_β are constant coefficients (both set as 1).

3.5 Team performance score (PS)

To quantitively assess the task completion quality and completion accuracy, we define the Performance Scores (PS) to grade the final configuration of the two assembled boxes. It serves as a metric for evaluating task success. In this study, the precision of the assembly can be assessed by the final configuration of the system, which can be represented by angles α_T and β_T as shown in equation (14). Both angles can be detected through sensors positioned on the Dynamic Box. These two angles are chosen because they fully characterize the spatial relationship between the two boxes and serve as intuitive and interpretable indicators of assembly accuracy [22].

$$Final\ configurations = < \alpha_T, \beta_T > \quad (14)$$

where T is the termination episode.

Therefore, the PS can be formulated as in equation (15), which not only offers a quantitative evaluation, but also aligns with the reward function [22], reflecting the extent to which the agent team aligns with the desired system performance criteria.

$$PS = 100 \cdot exp(-|\alpha_T - \alpha_{goal}|) \cdot exp(-|\beta_T - \beta_{goal}|) \quad (15)$$

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

By setting a scaling coefficient of 100, the PS is normalized within a 0 to 100 range. A score of 100 indicates a perfect execution of the task, demonstrating a perfect "L" configuration. This metric measures the team's overall performance, while also enabling further analysis of the training outcomes by predefining thresholds.

3.6 Role division and role scores

In this assembly task, agents contribute to the movement of the Dynamic Box by exerting forces that drive both linear translation and rotational motion. To systematically define and classify agent roles, we analyze their contributions to force and torque on the Dynamic Box. This allows us to categorize agents based on their functional roles and their impact on the system's motion. Thus, roles in this task are defined as:

- **Pushers:** Agents that apply force to maximize linear motion of the box, contributing to movement.
- **Rotators:** Agents that generate torque to induce rotation, contributing to box's orientation.
- **Adaptors:** Agents that frequently switch roles, adjusting their function dynamically.

We design roles scores to quantitively measure an agent's contribution to pushing, rotation, and adaptation, analyzing its specializes in a particular role, The scores are set as Pusher Score, P_{score}, Rotator Score R_{score}, and Adaptor Score, A_{score}, respectively.

Since these scores should enable meaningful relative comparisons between individual contribution with the team's preference, we firstly calculate the net force F_{net} and net moment M_{net} resulted by the entire team indicating the group inclination. Then, the scoring formulas are defined as shown in equations (16) – (18), allowing to assess whether each individual agent aligns with the team's overall motion strategy.

$$P_{score} = \frac{F_{net}}{\max(F_{net})} \quad (16)$$

$$R_{score} = \frac{|M_{net}|}{\max(|M_{net}|)} \quad (17)$$

$$A_{score} = 1 - \left|\frac{F_{net} + |M_{net}|}{F_{net} - |M_{net}| + \epsilon}\right| \quad (18)$$

where ϵ is a small value to prevent division by zero. It is set as 10^{-6} in the calculation.

In order to properly scale and ensure that the Adaptor should not be dominant over other roles, we conduct min-max normalization on $\mathrm{A_{score}}$. The penalty is applied as below:

$$A_{max} = \max(A_{scores}) \quad (19)$$

$$A_{min} = \min(A_{scores}) \quad (20)$$

$$A_i = A_{score} * (1 - 0.5\,P_{score} - 0.5R_{score}) \quad (21)$$

$$A_{score,scaled} = \frac{A_i - A_{min}}{A_{max} - A_{min} + \epsilon} \quad (22)$$

Through this way, the updated Adaptor Score, $\mathrm{A_{score,scaled}}$, is close to 1 if force and moment are similar, while this value is close to 0 if either Pusher or Rotator dominates. This ensures relative comparison wherein all scores are meaningful in a relative sense.

3.7 K-means clustering and feature extraction

We employ K-means clustering for role division, which is an unsupervised learning algorithm that partitions agents into distinct groups based on their action choices and other systematical factors. It is hoped to provide a data-driven results to identify and reveal role division pattern.

Given a dataset of n observations, K-means method partitions them into K clusters by solving the following optimization problem, shown in below equation:

$$\arg\min \sum_{i=1}^{K} \sum_{x_j \in C_i} \left|\left|x_j - \mu_i\right|\right|^2 \quad (23)$$

where $\mathrm{C_i} = \{\mathrm{C_1}, \mathrm{C_2}, \ldots, \mathrm{C_K}\}$ represents the set of K clusters. $\mathrm{x_j}$ is a data point (agent's motion contribution features), $\mathrm{x_j}$ is the data point (agent's motion contribution features), μ_i is the centroid of cluster $\mathrm{C_i}$.

Feature extraction: to effectively cluster agents based on their motion contributions, we extract a feature set that captures each agent's action patterns and overall impact on task performance. The feature vector for agent i is defined as:

$$X_i = [Frq(a_1), Frq(a_2), Frq(a_3), Frq(a_4), Frq(a_5), Frq(a_6), Steps, PS] \quad (24)$$

where $Frq(a_k)$ represents the normalized frequency of action a_i, computing by directly counting the number of times an agent selects a particular action and dividing it by the total number of steps taken. And "Steps" denotes the total number of steps completed by the team. PS refers to the value of Performance Score (PS), shown in equation (15).

By calculating and normalizing the action frequencies, we ensure that the clustering process remains scale-invariant. The inclusion of "Steps" and PS provides additional contextual information, enabling the clustering algorithm to consider more comprehensive systematical situations beyond solely physical implication.

4. RESULTS

4.1 Training results

In this study, the training is conducted using 10-agent teams, where the size is validated as effective through pre-trainings and based on our previous work results [22] [27] [28]. To reduce biases

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

introduced by initial conditions, random seeds are set ranging from 1-20.

Since the task goal is to achieve precision assembly through group collaboration, we evaluate training performance based on PS achieved by the entire team. Besides, the number of steps taken is also plotted in the figure to indicate the efficiency of learned policies. Thus, the training results are shown in Figure 5 where the x-axis represents the team index associating with a specific random seed (1-20). And in the y-axis, the blue dashed dots indicate PS values, and the bar plots represent the number of steps taken. The color of the bars indicates successful cases (green, PS > 55) and failure cases (res, PS < 55). This visualization provides insights into the relationship between team performance and task efficiency in this learning framework.

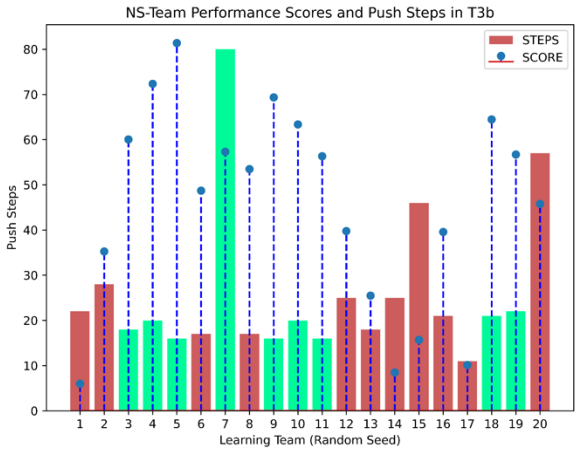


Figure 5: *Team performance scores (PS) and number of steps*

In Figure 5, the training results reveal a correlation between PS and the number of steps taken. In general, teams that achieve higher PS tend to complete the task in fewer steps, indicating greater efficiency in executing learned policies. It highlights the importance of team organization and role specialization in determining overall task performance.

Focusing on the successful cases (green bars), variations in efficiency can be observed. Some teams complete the task in fewer steps, indicating well-structured teamwork thus optimized coordination, while others required more steps, suggesting less efficient but still functional team dynamics. This variation shows that teams may adopt distinct organizational strategies, leading to differences in execution efficiency.

Moreover, the successful cases exhibit a relatively consistent number of steps, suggesting an optimal teamwork coordination pattern. Despite variations in learning dynamics, the fact that many successful teams converge to a similar range of steps implies that there exists an underlying coordination structure that leads to efficient task execution.

Those observations highlight the importance of understanding the effective role division and cooperation strategies for designing a learning-based multiagent system.

4.2 Role analysis from physical motion effect

In order to investigate the underlying mechanism of role division, we conduct physical analysis through examining the physical effects of each agent's action choices. Specifically, we analyze their actions via assessing whether agents' behavior align with optimal team strategies for executing the precision assembly tasks. This can aid in understanding how individual contributions influence the overall system dynamics.

Table 3 presents the physical effects of different actions in action space. As introduced in Section 3.3, each action a_i can result in forces in x and y directions and potentially contributing to rotation. Specifically, actions a_1, a_2, a_3, and a_4 generate x-direction forces and moments, while a_5 and a_6 introduce y-direction forces.

Table 3: *Physical effects of different actions*

	a_1	a_2,	a_3	a_4	a_5	a_6
F_x	-1	-1	1	1	0	0
F_y	0	0	0	0	1	-1
$Moment$	1	-1	1	-1	0	0

Given these physical constrictions, each agent is graded on their contribution towards moving the box. Each agent has 3 metrics that kept track of how much force they were exerting in the x and y direction, as well as how much they contributed to the moment individually. The net forces and net moments of all the agent's actions are then taken across all time steps. The net force in x and y were taken individually, then consolidated using the following equation:

$$F_{Net} = \sqrt{F_x^2 + F_y^2} \qquad (25)$$

Each individual agent's actions are then compared to the net values of force and moment. A contribution score was assigned wherein the agent gained +1 point if their force was positive and the net force was positive, -1 point if the net force was positive and the agent's action was negative. This logic was applied across all time steps and net values.

These scores were then compared to the net values in the original table to create a pushing score and moment score.

$$M_{score} = \frac{M_{Net}}{M_{a_n}} \qquad (26)$$

$$F_{score} = \frac{F_{Net}}{F_{a_n}} \qquad (27)$$

Where M_{a_n}, F_{a_n} represents the moment and force of the n agent across all time steps, and M_{Net}, F_{Net} are the net moment and forces of all agents in a specific timestep. The score was calculated for each time step.

The adaptor score was created by dividing the moment score by the pusher score.

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

$$Adapter\ Score = \frac{M_{score}}{F_{score}} \qquad (28)$$

Such that a greater adapter score is achieved by having similar moment and force scores.

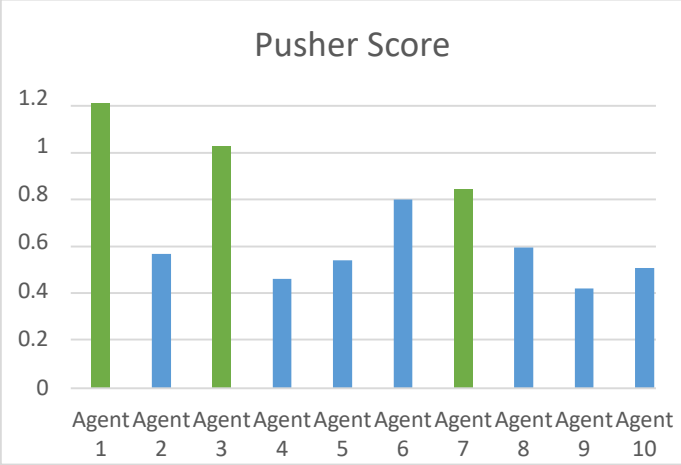


***Figure 6**: Pusher scores for the best team*

From Figure 6 we can see 3 agents in particular contribute more towards the pusher score, with the threshold being any agent scoring over 0.8. It is worth noting as well that agent 6 came very close to achieving this threshold.

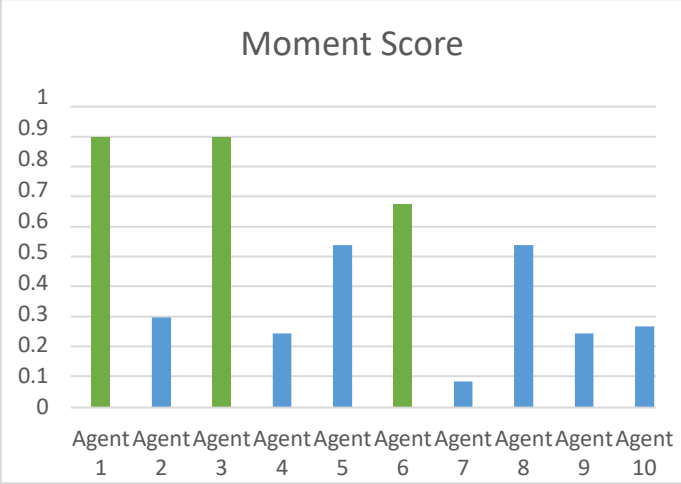


***Figure 7**: Moment score for the best team*

Figure 7 illustrates that Agent-1, 3, and 6 contribute the most to the rotation of the box. An interesting pattern that emerges shows that agents 1 and 3 contributed the highest scores for both moment and pusher scores.

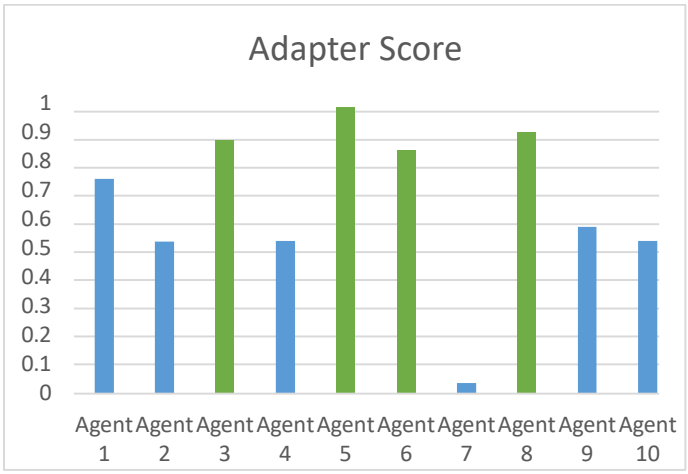


***Figure 8**: Adapter score for the best team*

From this data we can see there emerges a pattern of agents that score more in both fields and agents that consistently scoreless in both. This could reveal that there is an unbalanced workload between these agents, with Agent-2 and Agent-4 being examples of this.

The physical role analysis provides an unbiased assessment of work division and grounded in fundamental motion principles. This analysis serves as a crucial validation step for the following data-driven analysis through K-means method.

4.3 Role analysis through K-means in best team

Following the physical analysis, it is evident that role division emerges naturally as agents specialize in different physical motion roles. However, this method has certain task-specific limitations, and it only considers physical motion effects without incorporating higher-level behavioral patterns or adaptive decision-making observed in training dynamics.

Therefore, to conduct a more comprehensive analysis from a data-driven perspective, we apply the k-means clustering method to classify agent roles based on their action distributions and task contributions. This method allows us to capture both physical influences and behavioral tendencies. As a preliminary step, we perform clustering on the team that earned the highest performance scores. This ensures that the observed role patterns correspond to an effective teamwork strategy.

Guided by the findings from the physical analysis, we set the number of clusters to three. The P_{score}, R_{score} and A_{score} for each cluster are listed in below table.

***Table 4:** Clustering results*

Cluster	P_{score}	R_{score}	A_{score}	*Role*
0	0.72111	0.00	0.000003	Pusher
1	1.00000	1.00	0.000000	Pusher-Rotator
2	0.40000	0.25	0.999996	Adaptor

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

Besides the scores, the role assignment for the agent clusters is also determined and listed on the last column Table 4. It can be seen that each agent exhibits unique behavioral characteristics:

- Cluster 0 is assigned the role of **Pusher** due to its high Pusher Score (0.72111) and negligible Rotator Score (0.0). This indicates that agents in this cluster exert strong linear pushing forces without contributing to rotational motion, meaning their primary function is to drive the Dynamic Box forward in a straight path.

- Cluster 1 is identified as a **Pusher-Rotator** hybrid, with maximum Pusher and Rotator Scores (both 1.00). This suggests that agents in this cluster exert forces in a way that simultaneously pushes the Dynamic Box forward while also generating torque to induce rotation. These agents likely act to maximize both linear motion and rotational movement, making them critical for precise orientation adjustments.

- Cluster 2 is classified as an **Adaptor**, characterized by a moderate Pusher Score (0.40), moderate Rotator Score (0.25), and the highest Adaptor Score (0.999996). This indicates that agents in this cluster frequently switch roles between pushing and rotating, dynamically adjusting their contributions to stabilize the system.

The cluster results can be shown as below:

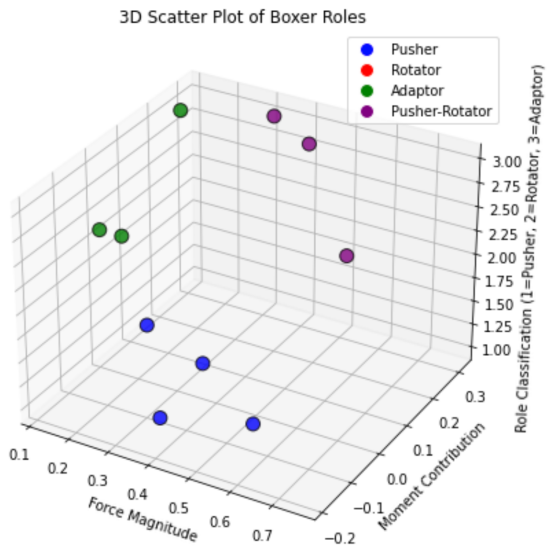


Figure 9: *3D scatter plot of agent roles*

We plot the role division in term of each agent's Force Magnitude and Moment Contribution. It is clearly shown that distinct roles emerge naturally aligning with the task objective. Despite all agents being homogeneous at the beginning, they gradually specialize in different functions.

4.4 K-clustering analysis in all groups

Building on the physical role and preliminary clustering analysis results, we extend the K-means clustering to all successful teams of agents to ensure their effective role specialization. Since all teams are trained under the same environment with the only difference being the random seeds, it is reasonable to assume that the emergent roles should be comparable across different teams.

Before performing clustering, we apply Principal Component Analysis (PCA) to examine the dominant features in the dataset and determine the key dimensions contributing to role differentiation. The results are shown in below figure, where the first eight principal components are extracted to assess their cumulative explained variance.

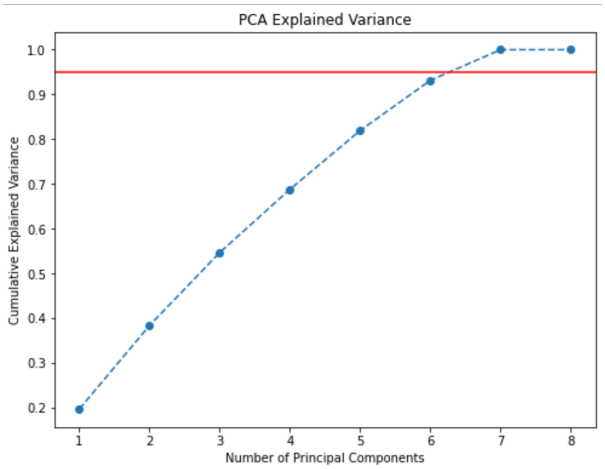


Figure 10: *Principal component analysis results*

The cumulative variance results are presented in Table 5. The first four components collectively account for over 70% of the total variance. Specifically, the first principal component (PC1) explains approximately 19.65% of the variance, followed by PC2 (18.68%), PC3 (16.31%), and PC4 (14.13%). The cumulative variance plot confirms that these four principal components retain the majority of the essential information from the original dataset while reducing its dimensionality.

Table 5: *Cumulative variance*

PC	Cumulative Variance
1	*0.1965*
2	*0.3833*
3	*0.5464*
4	*0.6877*
5	*0.8191*
6	*0.9313*
7	*1*
8	*1*

Based on the PCA analysis, the clustering result are shown in the below figure. We apply 3D scatter plot to visualize the results

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

based on PC1, 2, and 4 providing insight into the emergent role division among agents. In the figure, each point represents an individual agent. The distribution of agents suggest that the clustering successfully differentiates role-specialized behaviors, as agents naturally group into well-defined clusters rather than being randomly dispersed.

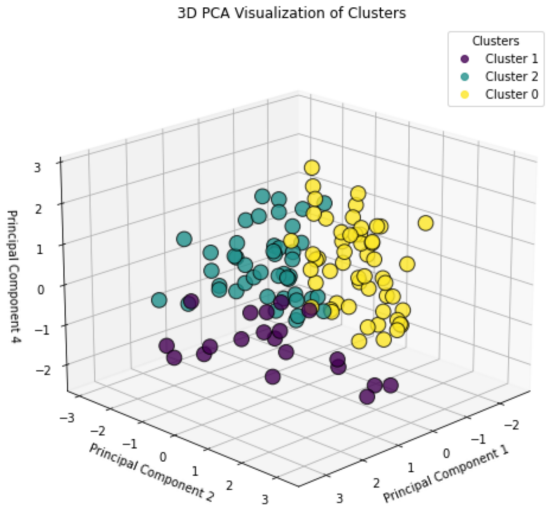


Figure 11: *3D scatter plot of clustering*

Based on the computed results and corresponding equations, the scores for each cluster are presented in the table below.

Table 6: *Clustering results*

Cluster	P_{score}	R_{score}	A_{score}	*Role*
0	0.077351	0.041962	0.999999	Adaptor
1	1.000000	1.000000	0.000000	Pusher-Rotator
2	0.713242	0.965671	0.000000	Rotator

The Pusher Score, Rotator Score, and Adaptor Score serve as indicators to quantify these contributions, allowing for a systematic interpretation of emergent roles within the agent team.

- Cluster 0 is assigned as **Adaptor**: This cluster has the highest Adaptor Score (0.999999), while maintaining relatively low Pusher (0.077) and Rotator (0.041) Scores. These agents act as stabilizers, dynamically adjusting their behavior based on task demands rather than committing to a specialized function. Their primary role is to balance the system, ensuring smooth transitions between pushing and rotating forces by adapting to their surroundings and the movements of other agents.
- Cluster 1 is identified as **Pusher-Rotator**. This cluster exhibits maximum values in both Pusher and Rotator Scores, indicating that these agents play a dominant role in both pushing and rotating. They likely provide the strongest forward force while simultaneously generating significant torque, ensuring effective movement and reorientation of the Dynamic Box.
- Cluster 2 is set as **Rotator**: Agents in this cluster have a high Rotator Score (0.965) along with a moderate Pusher Score (0.713), indicating a specialization in rotation with some degree of pushing. Unlike Cluster 1, these agents primarily facilitate rotation rather than drive linear movement, suggesting that they fine-tune the orientation of the Dynamic Box rather than directly controlling its trajectory.

This clustering result highlights the self-organized division of labor among agents, where different clusters take on specialized roles to achieve efficient motion coordination. The next section delves deeper into the interpretation of such behavior patterns.

5. DISCUSSION

5.1 Role division analysis

In order to offer a clear representation of how agents self-organize into distinct functional groups, we plot the results from Figure 12 into 3D scatter based on force magnitude, moment contribution, and role classification. The clusters, colored as Pusher (blue), Rotators (red), Adaptors (green), and Pusher-Rotators (purple), demonstrate these role division patterns within the multi-agent system.

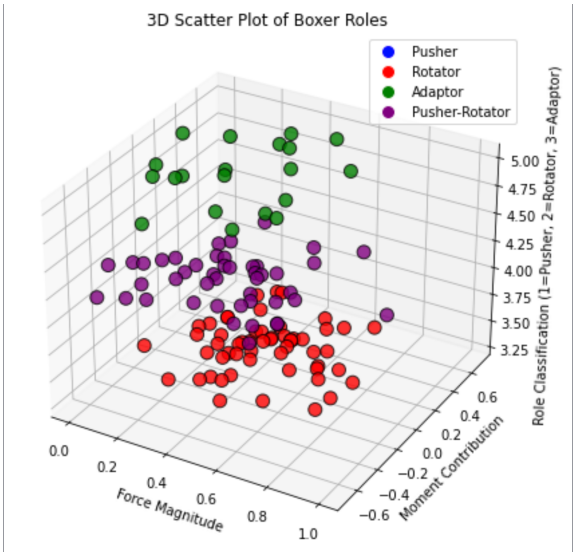


Figure 12: *3D scatter plot of role division*

5.1.1 Uneven workload distribution in emergent role specialization

It can be seen that Cluster-1, Pusher-Rotators, contribute significantly to both force and moment, indicating that they handle both movement and rotational control. While Cluster-2, Rotators, display high moment contribution. Cluter-0 Adaptors are distributed across these role boundaries. The results indicate that role differentiation is a natural outcome of the training process,

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

despite all agents starting with homogeneous capabilities. The presence of distinct roles ensures efficient task execution, as each agent can maximize its contribution rather than performing redundant or conflicting actions. Such mechanisms support the concept of self-organizing behaviors.

To further examine the workload, we plot cluster-wise force and moment contribution into bar chart, seen in Figure 13. It provides a quantitative breakdown of force magnitude and moment contribution for each role cluster.

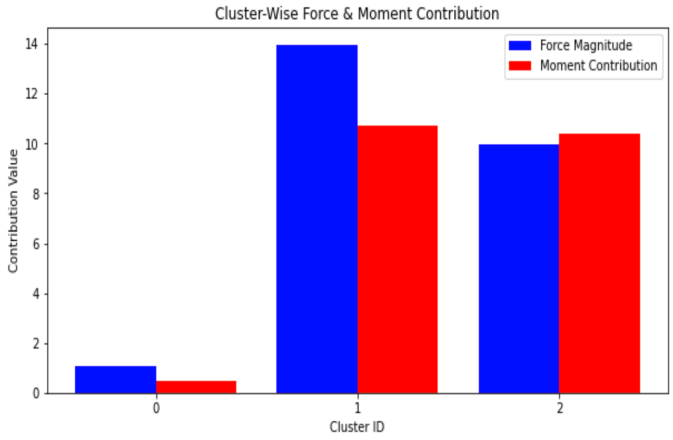


Figure 13: *Cluster-wise force and moment contribution*

It can be clearly seen that the workload among roles is not evenly distributed. Some agents bear significantly higher responsibility for task execution than others. In particular, the Cluster-1, Pusher-Rotators, contribute the highest force and moment values, indicating that they play the most critical role in driving both linear motion and rotational control. This suggests that a subset of agents consistently takes on the most physically demanding responsibilities.

In contrast, agents in Cluster 0, Adaptor, have minimal force and moment contributions, indicating that they are not primarily responsible for driving movement. Instead, they likely serve a different function in the system, which is explored further in the following section.

5.1.2 Hierarchical role division structure

Results shown in Figure 12 and Figure 13 indicate that role division follows a hierarchical structure rather than a parallel one. It can be observed that Cluster 1 (Pusher-Rotators) occupy the primary role, contributing both highest force and moment values and making them the key drivers of successful assembly. Cluster-2 Rotators function as secondary contributors, refining the orientation of the Dynamic Box rather than generating movement. They play a supportive yet crucial role in ensuring that the system maintains precision during assembly. Cluster 0-Adaptors hold the lowest position in the hierarchy, as their contributions are minimal.

This study shows that the MARL training would result in forming a structured system where different agent groups have varying impact on task execution. Effective teamwork in MARL does not require equal participation from all agents, but rather an optimized role distribution. Some agents assume more dominant and task-critical roles, while others act in supporting capacities.

5.1.3 Dynamic and adaptive nature of Adaptor role

Beyond role classification, the results also provide deeper insights into how agent roles emerge and contribute to overall system efficiency. A particularly interesting observation is the role of Adaptors in the system. Unlike Pushers and Rotators, whose functions remain relatively stable, Adaptors exhibit high variability in their contributions.

Shown in Figure 13, rather than committing to force application or rotational control, Adaptors adjust their actions in response to the needs of the team. These results indicate that their primary function is not to generate movement, but rather to stabilize the system.

What's more, the distribution of Adaptors shown in Figure 12 across role boundaries suggests that role transitions occur dynamically, allowing agents to adapt their strategies in response to environmental changes. Such functions contribute the system to remain flexible, adaptive, and resilient. It is reasonable to deduct that the presence of Adaptors plays key role for successful self-organization. They not only ensure the system maintains efficiency but also help the role assignments evolve based on real-time decision-making.

5.2 Interaction with Team Performance Scores (PS)

After purely focusing on the role division, we continue to analyze how such role division patterns interact with team performance. Thus, we plot the role assignments weighted by PS for each agent across different teams in Figure 14. In the plot, the x-axis represents the team index, and the y-axis represents the roles. This heatmap assists in revealing distinct role specialization strategies and their potential effects on task success.

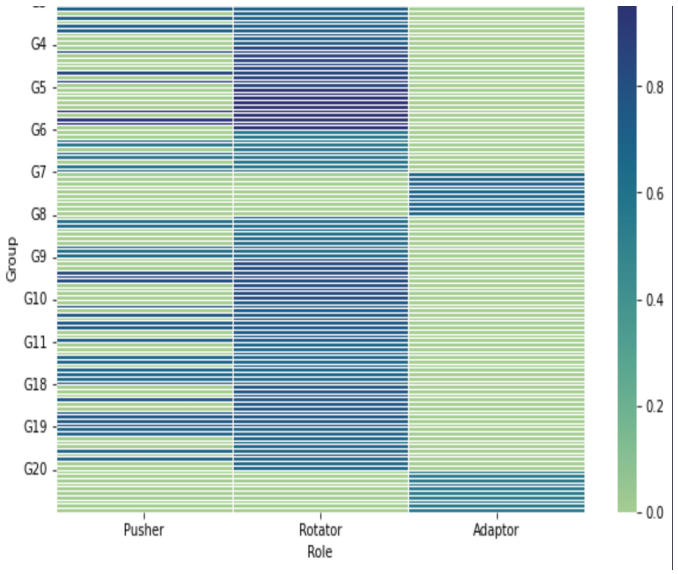


Figure 14: *Agent role assignments weighted by scores*

5.2.1 Distinct role strategies in teams

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

The results indicate that different teams prioritize different role distributions in their strategies. Some groups, such as G5, G6, and G8, heavily emphasize Rotators, suggesting that their primary approach relies on controlled rotation-based movement. These teams likely focus on orienting the Dynamic Box before applying direct forces for movement. Conversely, teams like G11 and G19 rely more on Pushers, suggesting a preference for direct linear force application to transport the box. The presence of these two distinct strategic approaches implies that teams develop unique movement patterns to complete the task.

5.2.2 Role specialization correlates with high performance

A noticeable trend in the heatmap is that teams with clear role specialization tend to achieve higher scores. It can be observed that G5 and G6, which likely employ a rotation-first strategy, ensuring precise control before executing the final assembly. On the other hand, G9 and G11, which favor Pushers, rely on straight-line force application to complete the task. Those teams achieve higher scores. In contrast, teams with weaker role differentiation, such as G7 and G20, appear to have a poor role structure, potentially leading to less efficient coordination and increased task completion time. This suggests that a strong work division may be a key factor in achieving optimal performance.

5.2.3 Adaptors are rare across all teams

It can be observed the relative absence of Adaptors across most teams. This suggests that while Adaptors provide stabilization and flexibility, they are not the primary drivers of the tasks. However, it is also shown that teams that do incorporate Adaptors tend to distribute them in small but strategic amounts. It indicates that Adaptors may still serve a crucial but limited role in balancing system dynamics.

6. CONCLUSION AND FUTURE WORK

This study investigates the emergence of role specialization in self-organized systems (SOSs) through multiagent reinforcement learning (MARL). By conducting the physical motion and K-means clustering analyses, we provide insights that drive self-organization in autonomous teams. The results contribute to a deeper understanding of the underlying mechanisms of role specialization, which will facilitate designing a more efficient system.

The experiments are conducted in a precision assembly task involving collision avoidance. Results shows that effective role differentiation can be emergent from MARL training process and result in a hierarchical structure. A key discovery is that agents naturally adopt specialized roles—such as pushers, rotators, and adaptors in this study—even without predefined role assignments. The presence of these emergent roles contributes to task success and learning efficiency. Additionally, the hierarchical nature of role division suggests that certain agents assume primary responsibilities in system movement, while others dynamically adjust their actions to stabilize coordination.

Future work will investigate the impact of social abilities within teams, such as communication and observation of peers, to uncover how these capabilities influence the formation and dynamics of role division. This work aims at bridging the gap between simulation-based results and real-world applications in the development of intelligent multiagent systems.

ACKNOWLEDGEMENTS

This paper is based on the work supported in part by the Autonomous Ship Consortium (ASC). The authors are grateful for their support.

REFERENCES

[1] U. Othman and E. Yang, "Human–robot collaborations in smart manufacturing environments: review and outlook," *Sensors,* vol. 12, p. 5863, 2023.

[2] N. Gjeldum, A. Aljinovic, C. Zizic and M. Mladineo, "Collaborative robot task allocation on an assembly line using the decision support system," *International Journal of Computer Integrated Manufacturing,* vol. 34, no. 4-5, pp. 510-526, 2022.

[3] M. K. Habib and Y. Baudoin, "Robot-assisted risky intervention, search, rescue and environmental surveillance," *International Journal of Advanced Robotic Systems,* vol. 7, no. 1, p. 10, 2010.

[4] H. Ji and Y. Jin, "Designing Self-Organizing Systems With Deep Multi-Agent Reinforcement Learning." International Design Engineering Technical Conferences and Computers and Information in Engineering Conference," *American Society of Mechanical Engineers,* p. Vol. 59278, 2019.

[5] C. E. Hrabia, M. Lutzenberger and S. Albayrak, "Towards adaptive multi-robot systems: Self-organization and self-adaptation," *The Knowledge Engineering Review,* vol. 33, p. e16, 2018.

[6] C. Chen and Y. Jin, "A behavior based approach to cellular self-organizing systems design," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2011.

[7] J. Hu and M. P. Wellman, "Multiagent reinforcement learning: theoretical framework and an algorithm," *ICML,* vol. 98, pp. 242-250, 1998.

[8] A. e. a. Wong, ""Deep multiagent reinforcement learning: Challenges and directions."," *Artificial Intelligence Review,* vol. 56, no. 6, pp. 5023-5056, (2023).

[9] B. Gerkey and M. J. Mataric, "A formal framework for the study of task allocation in multi-robot systems," *International Journal of Robotics Research ,* vol. 23, no. 9, pp. 939-954, 2004.

[10] M. Dorigo, V. Trianni, E. Sahin, R. Grob, T. H. Labella, G. Baldassarre and S. Nolfi, "Evolving self-organizing behaviors for a swarm-bot," *Autonomous Robots,* vol. 17, no. 2, pp. 223-245, 2004.

[11] T. Wang, G. Tarun, M. Anuj, P. Bei, W. Shimon and Z. Chongjie, "Rode: Learning roles to decompose multi-

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME

agent tasks." *arXiv preprint,* vol. arXiv:2010.01523, 2020.

[12] T. Wang, D. Heng, L. Victor and Z. Chongjie, "Roma: Multi-agent reinforcement learning with emergent roles." *arXiv preprint,* vol. arXiv:2003.08039, 2020.

[13] W. Chen, S. Yusheng, Z. Jingwei, Y. Cheng, Y. Chenfei, Q. Chen and C. Chi-Min, "Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents." *arXiv preprint,* vol. arXiv:2308.10848 2, no. no. 4, p. 6, 2023.

[14] T. Taniguchi, T. Kazuma and S. Tetsuo, "Role differentiation process by division of reward function in multi-agent reinforcement learning." *2008 SICE Annual Conference,* pp. 387-393, 2008.

[15] T. Guo, C. Xiuying, W. Yaqi, C. Ruidi, P. Shichao, N. V. Chawla, W. Olaf and Z. Xiangliang, "Large language model based multi-agents: A survey of progress and challenges.," *arXiv preprint,* no. arXiv:2402.01680, 2024.

[16] H. Zhu, "Agent Categorization With Group Role Assignment With Constraints and Simulated Annealing," *IEEE Transactions on Computational Social Systems,* vol. 7, no. 5, pp. 1234-1245, 2020.

[17] K. Ndousse, E. Douglas, L. Sergey and J. Natasha, "Emergent social learning via multi-agent reinforcement learning." *International conference on machine learning, PLMR,* pp. 7991-8004, 2021.

[18] G. Chen, "A New Framework for Multi-Agent Reinforcement Learning--Centralized Training and Exploration with Decentralized Execution via Policy Distillation," *arXiv preprint arXiv:1910.09152,* 2019.

[19] L. Buşoniu, R. Babuška and B. D. Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in multi-agent systems and applications-1,* pp. 183-221, 2010.

[20] D. Sam, D. Kudenko and M. Grześ, "An empirical study of potential-based reward shaping and advice in complex, multi-agent systems," *Advances in Complex Systems ,* pp. 251-278, 2011.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare and A. Graves, "Human-level control through deep reinforcement learning," *nature,* pp. 529-533., 2015.

[22] B. Huang and Y. Jin, "Reward shaping in multiagent reinforcement learning for self-organizing systems in assembly tasks," *Advanced Engineering Informatics,* p. 101800, 2022.

[23] B. Huang and Y. Jin, "Knowledge Transfer in Self-Organizing Systems and Impact of Social Ability," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,* vol. 87318, p. V03BT03A004, 2023.

[24] S. Ratchev, and N. Lohse, "Data modelling for web enabled design of modular precision assembly devices," *Assembly Automation ,* vol. 24.1, pp. 63-70, 2004.

[25] P. Shinners, , "Pygame-Python Game Development," *http://www.pygame.org," 2011..*

[26] M. Wunder, M. L. Littman and M. Babes, "Classes of multiagent q-learning dynamics with epsilon-greedy exploration," in *In Proceedings of the 27th International Conference on Machine Learning*, 2010.

[27] B. Huang and Y. Jin, "Social learning in self-organizing systems for complex assembly tasks," *Advanced Engineering Informatics,* vol. 57, p. 102109, 2023.

[28] B. Huang, H. Ji and Y. Jin, "Impact of Social Learning on Teamwork Efficiency in Learning-Based Self-Organized Systems," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,* vol. 86377, p. V03BT03A002, 2024.

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89206/V02AT02A057/7557301/v02at02a057-detc2025-169584.pdf by University of Southern California user on 03 November 2025

Copyright © 2025 by ASME