IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Scalable Multi-Agent Reinforcement Learning Framework for Multi-Machine Tending

Abdalwhab Bakheet Mohamed Abdalwhab[1], Giovanni Beltrame[2], David St-Onge[1]

***Abstract*— Robotic manipulators hold significant untapped potential for manufacturing industries, particularly when deployed in multi-robot configurations that can enhance resource utilization, increase throughput, and reduce costs. However, industrial manipulators typically operate in isolated one-robot, one-machine setups, limiting both utilization and scalability. Even mobile robot implementations generally rely on centralized architectures, creating vulnerability to single points of failure and requiring robust communication infrastructure.**

This paper introduces SMAPPO (Scalable Multi-Agent Proximal Policy Optimization), a scalable input-size invariant multi-agent reinforcement learning model for decentralized multi-robot management in industrial environments. MAPPO (Multi-Agent Proximal Policy Optimization) represents the current state-of-the-art approach. We optimized an existing simulator to handle complex multi-agent reinforcement learning scenarios and designed a new multi-machine tending scenario for evaluation. Our novel observation encoder enables SMAPPO to handle varying numbers of agents, machines, and storage areas with minimal or no retraining. Results demonstrate SMAPPO's superior performance compared to the state-of-the-art MAPPO across multiple conditions: full retraining (up to 61% improvement), curriculum learning (up to 45% increased productivity and up to 49% fewer collisions), zero-shot generalization to significantly different scale scenarios (up to 272% better performance without retraining), and adaptability under extremely low initial training (up to 100% increase in parts delivery).

I. Introduction

Robotic technologies, especially those equipped with artificial intelligence [1], still have great untapped potential for the manufacturing industry. Robots excel at repetitive, labor-intensive tasks, freeing human workers to focus on more creative activities. Moreover, deploying multiple robots in collaborative systems promises enhanced resource utilization, increased throughput, and cost efficiency. However, implementing multi-robot systems introduces considerable complexity in terms of communication, coordination, and collision avoidance. Standard industrial solutions typically rely on a centralized server for trajectory planning and task coordination, ensuring efficient execution and collision-free operation. While effective, these centralized methods suffer from limitations such as single points of failure and heavy reliance on robust, continuous communication [2].

Alternatively, decentralized multi-agent control offers increased flexibility, allowing robots to make localized, independent decisions based on real-time environmental observations. In this context, Multi-Agent Reinforcement Learning (MARL) emerges as a promising solution, enabling robots to learn collaboratively and autonomously through environmental interactions.

Given that resilience is a core driving value in Industry 5.0 [1], scalable robotic systems capable of rapid adaptation have become essential. Traditional reinforcement learning (RL) and MARL frameworks require extensive training times and operate on fixed input dimensions. Consequently, scaling up or down—such as adding or removing machines and robots—usually necessitates retraining from scratch, leading to considerable inefficiency in terms of time, cost, and effort.

This work introduces a scalable, input-size invariant MARL framework to address the challenge of scaling robotic systems in manufacturing environments, minimizing or entirely eliminating the need for retraining. As a practical demonstration, we focus on machine tending–a common, repetitive manufacturing task typically performed by fixed robotic arms dedicated to individual production machines. Although effective, this conventional approach demands substantial human involvement, especially in transferring parts between machines, and faces significant scalability constraints, as each additional machine requires an extra robotic arm. Moreover, a single robot failure can halt the associated machine's production.

A more adaptive solution involves employing a swarm of autonomous mobile manipulators capable of decentralized coordination, independently navigating among multiple machines, assigning tasks, supplying raw materials, and managing finished parts. This scenario is ideally suited for the application of MARL techniques. Our goal is to transition MARL research from simplified simulations—such as Starcraft Multi-Agent Challenge (SMAC) [3], Google Football [4], and basic formation control and goal coverage tasks [5]–[7]– towards realistic, industry-oriented manufacturing scenarios. We propose a robust, input-size invariant MARL solution tailored specifically to decentralized mobile robots performing independent navigation and task allocation for multi-machine tending. Our main contributions are:

- Optimize an existing simulator to better support large-scale MARL benchmarking;
- Design a scalable multi-agent multi-machine-tending scenario;
- Develop a novel observation encoding technique that is input-size invariant, enabling scalability;
- Introduce a new scalable MARL model (SMAPPO);

We validate our framework through an extensive analysis of its performance and scalability with minimal or no retraining.

[1]Department of Mechanical Engineering, ETS Montreal, [2]Department of Computer and Software Engineering, Polytechnique Montréal.

*This project was funded by NSERC Alliance grant ALLRP 566182 - 21 and NSERC CREATE CoRoM program. We also acknowledge the support provided by Calcul Québec and Compute Canada.

II. Background and Related Work

A. RL for Machine Tending

Few researchers have explored RL specifically for machine tending tasks. Iriondo et al. [8] presented a system where a single mobile manipulator learned to optimally position itself to retrieve parts by using feedback from the manipulator's planner. Initially employing Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO), they later transitioned to Twin Delayed DDPG (TD3), demonstrating success in simulation and reliable performance on a physical robot [9]. Their work was restricted to a single robot tending a single machine and did not account for practical considerations like part availability, part dropping, or multiple collection trips.

For multi-agent and multi-machine settings, Agrawal et al. [10] developed a PPO-based method for multi-robot job scheduling and navigation within a manufacturing environment, considering processing delays and machine failures. Nevertheless, their model implicitly assumed infinite onboard storage capacity and relied on a centralized server for communication and marginal centralized control. In a more decentralized approach, Abdalwhab et al. [11] integrated an attention-based encoder with MAPPO, addressing both task assignment and navigation for machine tending. Although comprehensive, both works require a fixed observation size, limiting scalability and adaptability when system configurations change.

B. Scaling MARL Solutions

Multi-agent reinforcement learning (MARL) is an emerging field of study that aims to expand RL to multi-agent environments. Notable examples include MAPPO [12], an extension of PPO, and MADDPG [13], which extends Deterministic Policy Gradient (DDPG). Both methods leverage a centralized critic during training to leverage additional global information, omitting this centralized component during execution. These models do not inherently support rapid scalability: when the environment's agent or object count changes, models typically require retraining from scratch, presenting substantial challenges for real-world implementation.

Common strategies to circumvent input size limitations include padding observations with zeros or ones to maintain a constant input dimension [5], or imposing a maximum number of observable entities, ignoring any surplus [14], [15]. However, both strategies have drawbacks: padding introduces unnecessary data, complicating learning, while limiting observations discards potentially valuable information, harming performance.

To address these issues, Muning Wen et al. [16] introduced a centralized solution that predicts agents' actions sequentially using a transformer-based encoder-decoder. However, this approach suffers from scalability limitations due to the inherent drawbacks of centralized control and the computational overhead of sequential decision-making.

Some studies have addressed scalability in MARL using more sophisticated representation learning methods. Agarwal et al. [5] and Wang et al. [17] used Graph Neural Networks (GNN) to encode the environment information of variable size into a fixed-size vector.

Similarly, Qian Long et al. [6] introduced an observation-action encoder that assigns a distinct encoder and attention layer for each entity type (agent and landmark). It computes attention between agents, as well as between the agent and landmarks, integrating the result with the agent's action. Additionally, they incorporated a global attention layer to aggregate agents' observations and actions into the centralized Q-function of MADDPG. For the MADDPG actor, they used the same observation-action encoder but excluded the action encoding. Their design enabled the model to accommodate varying numbers of entities within fixed-size representations.

A typical evaluation for input-size invariant representation methods is to show efficient curriculum learning and possibly zero-shot transfer to new scenarios. For instance, Wang et al. [17] showed effective curriculum learning to gradually increase the number of agents that their model can handle, but did not experiment with zero-shot transfer. While Agarwal et al. [5] demonstrated effective zero-shot transfer for goal coverage tasks ($\pm$ 3 agents), and formation control ($\pm$ 1 agent). On the other hand, Qian Long et al. in [6] achieved zero-shot transfer for up to twice the original agent count in simulated food collection tasks.

Unlike these prior models, we propose a MARL framework specifically tailored for realistic manufacturing scenarios, focusing on decentralized mobile robots that independently handle dynamic task assignment, machine navigation, and part transportation. Our approach emphasizes scalability, supporting rapid system expansion or contraction with minimal or no retraining, thus aligning better with practical manufacturing requirements.

III. Problem Definition

Multi-agent, multi-machine tending is fundamentally a coordination problem involving N autonomous robots tending to M machines. Robots are responsible for supplying raw materials, collecting finished parts, and delivering them to K designated storage areas. When parts get ready at different machines, each robot independently selects tasks, navigates safely to pick up finished parts, and transports them to storage locations. Collisions between robots, machines, or storage zones must be avoided, requiring intelligent decision-making and trajectory planning. Upon successful collection by a robot, each machine requires a fixed processing duration T before the next part becomes available.

We acknowledge the importance of object orientation detection, grasping, and placement at the destination; however, we specifically focus on the scalability of multi-robot coordination, navigation, and event sequencing for up to 100 robots as core challenges. To emphasize these fundamental aspects, we assume robots perform pick-and-place operations without stopping, a desirable [18], achievable [19], and efficient approach as demonstrated in [20], where they achieved 48% task time reduction on a real robot. Additionally, we assume external management of raw material supply, thereby

isolating the coordination and navigation complexities central to scalable manufacturing operations.

IV. Methods

A. MAPPO backbone

MAPPO [12] is a multi-agent reinforcement learning algorithm designed for decentralized partially observable Markov decision processes (DEC-POMDP), characterized by $\{n, \mathcal{A}, \mathcal{S}, O, \mathcal{R}, \mathcal{P}, \gamma\}$. Where n is the number of agents, $\mathcal{A}$ is the shared action space, and $\mathcal{S}$ is the state space. Each agent i has a local observation $o_i = O(s, i)$ of the global state s. Agents select actions based on a policy $\pi_\theta(o_i)$, with θ as the learnable parameters of the policy, $\mathcal{P}(\acute{s}|s, A)$ is the probability of state transition from s to $\acute{s}$ given the agents' joint actions $A = (a_1, a_2, ..., a_n)$. $R(s, A)$ is the shared reward, and the objective is to maximize the expected sum of discounted rewards:

$$J(\theta) = \mathbb{E}_{A^t, s^t}[\sum_t \gamma^t R(s^t, A^t)] \quad (1)$$

where γ is the discount factor, giving preference to short-term rewards over long-term ones.

MAPPO is based on a paired-network architecture consisting of an actor (policy) and a critic. The policy selects actions to maximize accumulated discounted returns. The critic estimates the state value, which is used for variance reduction. Because the critic is not used during the execution, it can access the global state during training without violating the centralized training, decentralized execution (CTDE) principle.

B. Scalable MAPPO (SMAPPO)

We introduce a novel attention-based input-size invariant observation encoder and integrate it into both the actor and critic networks of MAPPO, resulting in a new scalable model, SMAPPO. This enhancement not only improves performance but also allows the model to handle varying numbers of agents, machines, and storage areas with minimum or no retraining.

Fig. 1 illustrates the enhanced actor network. Inspired by Long et al. [6], we designed a novel agent observation encoder. Flattening the entire observation into a single vector removes important relational and type-specific information about entities. To address this issue, we group entities by type—current agent, machines, storage areas, and other agents—and employ a separate encoder (one Fully Connected Layer) for each entity type, allowing the model to learn a different encoder for each entity type.

Each group of entities is stacked into a vector and processed through a multi-head attention mechanism with the agent's encoding, producing a fixed-size representation for each group, regardless of the number of entities.

Finally, these representations are concatenated with the agent's encoding to form an observation embedding that remains invariant to changes in the number of agents, machines, or storage areas.

The encoded observation is subsequently processed through the typical layers of an open-source implementation of MAPPO, consisting of a series of fully connected (FC) layers with Tanh activations, normalization layers (LayerNorm), a Gated Recurrent Unit (GRU), and a final FC to produce the action.

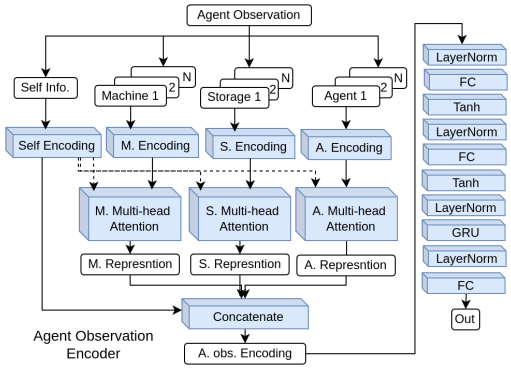

Fig. 1: SMAPPO actor network depicting the novel input-size invariant agent observation encoder

For the critic network (Fig. 2), which has access to the observations of all agents, we designed an encoder that aggregates these inputs into a fixed-size vector, regardless of the number of agents. Each agent's observation is first encoded using the same observation encoder design as in the actor, producing a consistent representation for each agent. These representations are then concatenated and passed through a multi-head attention module with a learnable query parameter to capture the global context in a fixed vector size. The attention output is subsequently processed through a series of layers mirroring those in the actor network, with the final layer generating the value estimate instead of the action.

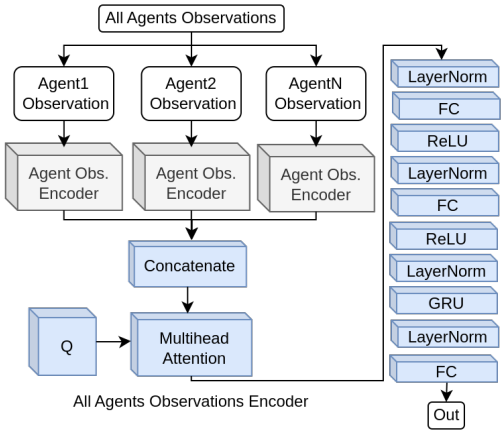

Fig. 2: SMAPPO Critic network demonstrating the novel input-size invariant encoding

See also the supplementary materials[1] for more details about the model specifications.

[1] https://initrobots.ca/smappo/

C. Observation and Reward Design

We adopted the observation and reward in [11], with each agent's observation consisting of:

- The agent's absolute position and a binary flag indicating whether it is currently carrying a part.
- The relative positions of all machines, each paired with a flag signaling if a part is ready for pickup.
- The relative positions of the storage areas.
- The relative positions of other agents, along with their individual "has part" status flags.

We retained all reward components from [11], except for the utilization penalty. To prevent it from scaling uncontrollably as the number of machines increases, we adjusted this term to ensure stability and maintain balanced learning. The reward components are as follows:

1. Base Rewards:

- Pick Reward (R_{pi}): Earned when an agent with no part arrives at a machine that has a ready part.
- Place Reward (R_{pl}): Granted when an agent successfully delivers a part to a storage area.
- Collision Penalty (R_c): Applied whenever the agent collides with other agents, machines, the storage area, or obstacles like walls.

2. Distance-based Rewards:

- Progress towards the closest machine with a ready part (R_{pm}) and the storage area (R_{ps}) is rewarded to provide continuous feedback. The rewards are calculated based on the reduction in distance to the target (d_m^t) between consecutive steps, scaled by a factor (s_{pr}).

$$R_{pm}^t = \begin{cases} s_{pr} * (d_m^{t-1} - d_m^t) & \text{agent has no part} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$R_{ps}^t = \begin{cases} s_{pr} * (d_s^{t-1} - d_s^t) & \text{agent has part} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

3. Machine Utilization Penalty (R_u):

- To minimize idle machine time, a penalty is assigned based on the number of uncollected parts (p_{un}) divided by the number of machines M (to be bounded between 0 and 1 for any number of machines). Then multiplied by a scaling factor (s_u)

$$R_u = s_u * p_{un}/M \tag{4}$$

4. Time Penalty (R_t):

- Applied to discourage stopping, particularly when no other rewards or penalties are being applied.

$$R_t = \begin{cases} tp & \text{if } R_{pi}, R_{pl}, \text{ and } R_u \text{ are } 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The total reward is the aggregation of all the above terms:

$$R_T = R_{pi} + R_{pl} + R_c + R_{pm} + R_{ps} + R_u + R_t \tag{6}$$

The reader may consult the supplementary materials for the tuned values of the constants in the reward components.

D. Simulation

For our experiments, we started with the Vectorized Multi-Agent Simulator (VMAS) [21], built for efficient benchmarking of multi-agent reinforcement learning (MARL). VMAS incorporates a 2D physics engine and offers compatibility with the popular gym-style simulation environments, facilitating efficient prototyping and rapid scenario customization through its modular and open-source architecture.

VMAS encodes the environment state—positions and velocities of objects— as continuous values, while agents' actions translate into physical forces along the x and y axes (f_x and f_y). The physics engine processes these forces alongside dynamics such as velocity damping and collision interactions, updating the environment's state accordingly [21].

As in [10], this work considers discrete holonomic actions (0 as no acceleration, 1 meaning leftward acceleration, 2 for rightward acceleration, 3 for downward acceleration, and 4 for upward acceleration). This is equivalent to a robot with holonomic velocity commands $\{v_x, v_y | v_x * v_y = 0, abs(v) = \{0, K_{speed}\}\}$, where K_{speed} is constant.

To support more complex scenarios with numerous agents and objects, we optimized VMAS. First, we redesigned the observation generation process, shifting from agent-wise sequential computation to a more efficient global observation approach. Instead of separately processing each agent's observation, the environment now processes them once per timestep, generating a global observation that is then tailored to each agent. This optimization notably reduces computational overhead, making complex scenarios feasible.

Similarly, reward calculation was restructured to minimize redundant computations by sharing common reward components calculated globally rather than repeatedly per agent. This design allows for improved efficiency ($(19 \pm 1.5)\%$ time reduction (details on the supplementary material)), particularly in dense-reward environments involving numerous agents and objects.

Then we designed the multi-agent, multi-machine tending scenario shown in Fig. 3. The environment includes N agents (shown in red), M production machines (in green), and K storage areas (in blue), all enclosed by black boundary walls. The dotted lines in the figure reference the physical counterparts intended for real-world deployment: our RanGen robot, CNC, and storage shelves. The scale of the number of entities to the environment size should be roughly the same for all experiments. For simplicity, instead of changing the room size when adding more entities, we kept the room size unchanged ($2.3m \times 2.3m$) and scaled the entities' sizes (details in the supplementary materials).

Each simulation episode runs for 200 timesteps, during which agents strive to maximize the number of parts collected and delivered. Agents are limited to carrying a single part at a time, requiring them to deliver the current part before picking up another. After each retrieval, machines require 20 timesteps to prepare subsequent parts. The 200-step horizon thus effectively tests agents' coordination, adaptability, and efficiency in dynamically changing operational states.

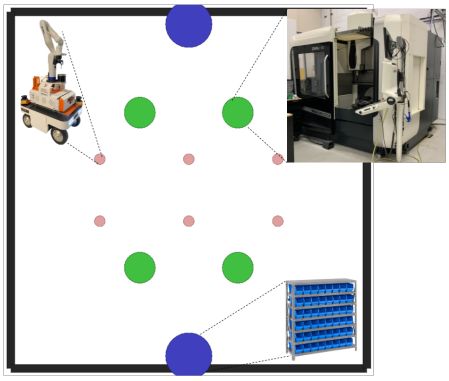

Fig. 3: Multi-agent (red) multi-machine (green) tending scenario designed in the optimized VMAS, featuring storage areas (blue). Dotted lines indicate the real-world counterparts planned for deployment: the RanGen robot, CNC, and storage shelves.

E. Evaluation Metrics

Our evaluation framework is designed to assess both task effectiveness and resource efficiency through various criteria:

1) Task Success Factor: Measured by the total number of delivered Parts by all robots.
2) Safety Factor: assessed by the total Number of Collisions: Calculated as the cumulative collisions across all agents, indicating the safety of navigation and interaction.
3) Resource Utilization Factor: Expressed by average machine utilization (MU):

$$MU = \frac{P_c}{P_{max}} \tag{7}$$

Where P_c is the total number of collected parts from all machines, P_{max} is the theoretical maximum number of parts that could have been produced by all machines (if they are not waiting for agents to pick ready parts).

Each experiment was repeated three times with different random seeds to ensure reliability, with the results presented as the mean values and standard deviations between different seeds.

V. Experiments

Due to the complexity of the task addressed by our solution—which integrates both task allocation and force-based navigation—it is not directly comparable to approaches that focus solely on scheduling (e.g., greedy algorithms) or employ standard navigation strategies (such as velocity-based Optimal Reciprocal Collision Avoidance). Consequently, we conducted an extensive evaluation against MAPPO, a state-of-the-art multi-agent reinforcement learning model, to provide a more meaningful and thorough performance comparison.

This section presents the four main experiments that were conducted, gradually evaluating the model's performance and scalability to new scenarios with minimum training. Firstly, full retraining from scratch at each new scenario, focusing only on assessing model performance in different scenarios. Followed by evaluating the ability to leverage previous learning experience from one scenario to the next (curriculum learning), and testing the ability to scale to new scenarios without any retraining (Zero-shot Evaluation). Finally, we evaluate the model's adaptability with curriculum learning under extremely low initial training. The experiments tested the model with varying numbers of agents (2-100), machines (2–28), and storage areas (2–14), with sampled values within these ranges due to computational limits.

A. Full Retraining

Based on the layout shown in Fig. 3, we designed 6 setups with increasing difficulty to compare SMAPPO to MAPPO (listed on the x-axis of Fig. 4). The first setup has 6 agents, 4 machines, and 2 storage areas. Then, each subsequent setup adds 2 more agents and, if needed, 2 more machines, and storage areas to maintain the original ratio. The final setup focuses on large-scale testing with 100 agents, 16 machines, and 2 long rectangular storage areas. Limited by computational resources, it was trained for 100 episodes (3.54 days per seed on an AMD EPYC 9654 CPU (more details in supplementary materials)). As shown in Fig. 5, the model can achieve about half of its learning within those 100 episodes.

Each model was retrained from scratch in each setup for 2,000 episodes, whenever reasonable with the available computational resources. The performance metrics were averaged over the final 100 episodes to assess stability and effectiveness.

This experimental setup inherently favors MAPPO due to MAPPO's input-size dependence; by default, it can not transfer the learning from one scenario to another scenario with a different number of entities. Thus, we trained a distinct MAPPO model for each scenario, adjusting the input layer sizes of both the actor and critic networks to suit the specific requirements of each case. In contrast, the same SMAPPO model was trained on all setups. Parts delivery and collisions are plotted in Fig. 4, and machine utilization is reported in Table I.

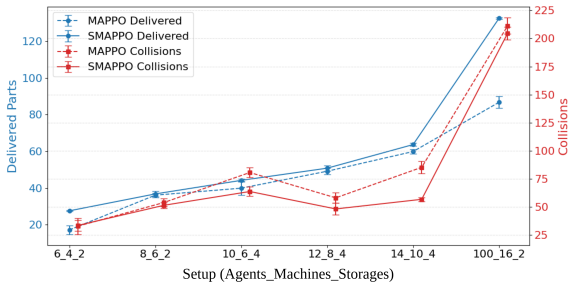

Fig. 4: Full retraining parts delivery and collisions average and std as error bars comparing the same SMAPPO model to setup-specific MAPPO models.

Even though the same SMAPPO model was trained across all setups, while a distinct MAPPO model was tailored for

each setup, the results reveal that SMAPPO consistently outperformed MAPPO in all setups. SMAPPO delivered more parts in all of the scenarios. Furthermore, as the scale of the experiment increased to involve 100 agents, the performance gap widened significantly, with SMAPPO demonstrating a substantial 53% improvement over MAPPO in parts delivery.

In terms of safety, SMAPPO demonstrated a lower number of collisions in all scenarios except the smallest setup, where the difference was still not significant. This proves that SMAPPO is relatively safer than MAPPO. Going against expectation, a drop in collisions was noticed when going from (10_6_4) setup to (12_8_4) despite having more entities in the environment. This can be mainly explained by the fact that in this setup, agents' size was reduced to accommodate the increase in the number of entities compared to the previous three setups. With fewer agents, MAPPO performs comparably to SMAPPO, but as the number of agents—and thus coordination difficulty—increases, MAPPO's performance degrades while our model's advantage becomes evident.

TABLE I: Full retraining machine utilization average and (std) comparing the same SMAPPO model to setup-specific MAPPO models, last setup was trained for 100 episodes.

Setup	MAPPO	SMAPPO
6_4_2	0.51(0.05)	**0.74(0.01)**
8_6_2	0.65(0.02)	**0.66(0.02)**
10_6_4	0.72(0.06)	**0.79(0.01)**
12_8_4	0.65(0.02)	**0.69(0.02)**
14_10_4	0.65(0.02)	**0.69(0.01)**
100_16_2	0.64(0.02)	**0.91(0.00)**

The same is noticed for utilization, where SMAPPO outperforms MAPPO in all setups. Also, as the number of agents, machines, and storage areas increases, SMAPPO tends to show way better utilization of machines, reaching a 91% utilization in the large-scale setup.

To further analyze the learning dynamics, Fig. 5 illustrates the total episode reward accumulated by all agents during training in the 14 agents, 10 machines, and 4 storage areas scenario. The curves represent the mean and standard deviation across three independent runs with different random seeds. From the early stages of training, it is evident that SMAPPO learns the task faster than MAPPO. In fact, SMAPPO reached 50% of its final performance in 100 episodes, and 70% in 200 episodes. Throughout the training process, a considerable performance gap is maintained between the two, highlighting the superior learning efficiency and capability of SMAPPO. We also experimented with 8,000 training episodes, where SMAPPO still outperformed MAPPO (see the supplementary materials for details).

B. Curriculum Learning

This section assesses SMAPPO's adaptability to dynamic changes in facility scale, specifically the incremental addition of agents, machines, and storage areas after initial training. The experiment begins with 2000 training episodes in a

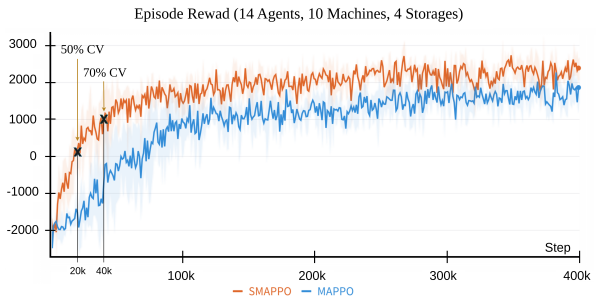

Fig. 5: Full retraining total episode reward: mean and standard deviation of three different seeds. Bars showing the 50% and 70% of the final convergence value (CV)

baseline configuration (6 agents, 4 machines, and 2 storage areas). Subsequently, every 100 episodes —the time it takes the model to reach 50% of its performance if it was trained from scratch— the environment is expanded by adding 6 agents, 4 machines, and 2 storage areas. This setup allows us to evaluate how efficiently the model leverages newly introduced resources and how such changes impact its overall learning performance.

By default, MAPPO does not support curriculum learning due to its requirement for a fixed input size -any change in the number of agents, machines, or storage areas alters the input size. To be able to compare MAPPO to SMAPPO, we implemented a zero-padding strategy for MAPPO. In this method, the observation is padded by zeros to reach the desired fixed input size. The evaluation results are presented in Fig. 6, and Table II with more details and visualization of the training provided in the supplementary materials.

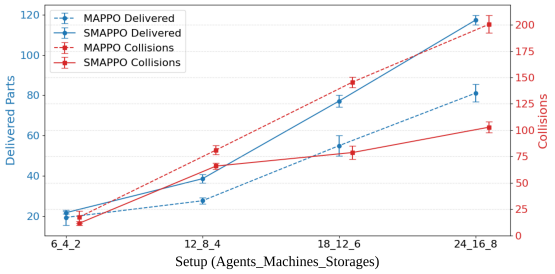

Fig. 6: Curriculum Learning parts delivery and collisions average and std as error bars.

TABLE II: Curriculum machine utilization average and (std)

Setup	MAPPO	SMAPPO
6_4_2	0.53(0.1)	**0.59(0.01)**
12_8_4	0.39(0.02)	**0.54(0.02)**
18_12_6	0.5(0.04)	**0.7(0.02)**
24_16_8	0.55(0.03)	**0.79(0.02)**

The results of the curriculum experiment show that SMAPPO can quickly adapt and benefit from the increase in agents, machines, and storage areas. In summary, SMAPPO delivered more parts (up to 45% increase) while considerably

reducing collisions in all setups (up to 49% decrease). Moreover, in terms of utilization, SMAPPO achieved superior performance in all setups.

Furthermore, by monitoring the total collected reward accumulated by all agents (as shown in Fig. 7), it becomes evident that the performance of MAPPO experiences a noticeable decline with each introduction of new entities at steps 400K, 420K, and 440K. After these introductions, MAPPO struggles to recover quickly, resulting in a limited ability to capitalize on the newly available resources. In contrast, SMAPPO demonstrates greater adaptability: its performance drops only during the first introduction of new entities but recovers rapidly thereafter. Notably, during the subsequent two phases of entity addition, SMAPPO directly benefits from the increased resources without any significant drop in performance, showcasing its superior resilience and efficiency in dynamic environments.

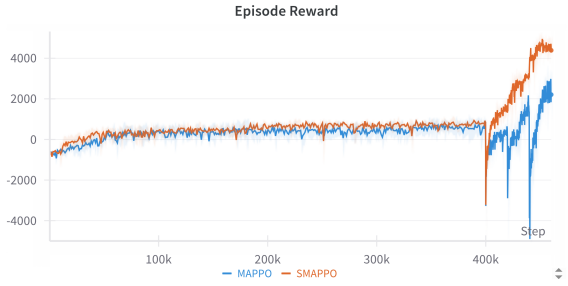

Fig. 7: Curriculum learning total episode reward reported as mean and standard deviation of three different seeds, new entities (agents, machines, and storage areas) were introduced at 400K, 420K, and 440K

This makes SMAPPO a more suitable option for real deployment because it can adapt to the increase in the manufacturing setup with little training.

C. Zero-shot Evaluation

SMAPPO is compared to MAPPO in terms of its ability to perform in a new setting that it has not been trained on. The new scenario will have fewer or more agents, machines, and storage areas. Again, zero-padding was adopted for MAPPO to allow it to run in the evaluation scenarios.

Each model (MAPPO and SMAPPO) was trained for 2000 episodes in the initial setup with 6 agents, 4 machines, and 2 storage areas. Then, the model weights were frozen and used for evaluation. For each new setup, a new environment instance is initialized with the correct number of agents, machines, and storage areas, and the model is evaluated for 50 episodes. Results are depicted in Fig. 8 and Table III and visualized in the supplementary materials online.

Fig. 8 shows that SMAPPO can greatly enhance parts delivery in all setups over MAPPO. For instance, for the large-scale setup (7 times more agents, machines, and storage areas than the training setup), SMAPPO achieved a 272% gain in parts delivery. The superiority of SMAPPO becomes clearer as the new setup goes far from the training setup.

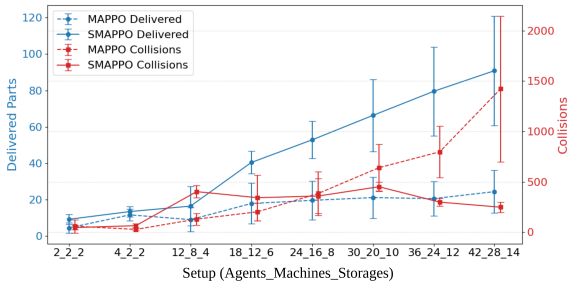

Fig. 8: Zero-shot parts delivery and collisions average and std as error bars.

With regards to collisions, SMAPPO demonstrated a safer behavior, making fewer collisions in setups that were far from training (1 setup with fewer objects, and 4 with more objects than in training). Whereas, MAPPO performed better in 3 setups that were close to training.

TABLE III: Zero-shot machine utilization average and (std)

Setup	MAPPO	SMAPPO
2_2_2	0.26(0.13)	**0.5(0.13)**
4_2_2	0.65(0.14)	**0.73(0.15)**
12_8_4	0.21(0.05)	**0.31(0.1)**
18_12_6	0.22(0.06)	**0.4(0.05)**
24_16_8	0.19(0.03)	**0.4(0.06)**
30_20_10	0.16(0.02)	**0.39(0.09)**
36_24_12	0.13(0.03)	**0.39(0.09)**
42_28_14	0.12(0.04)	**0.38(0.09)**

Similarly, for machine utilization (Table III), SMAPPO performed better in all setups, maintaining machine utilization of 39% or above even in setups that are very far from training. While MAPPO utilization drops as the scenario moves farther from the training.

Overall, those zero-shot results clearly show that SMAPPO can generalize better to new setups with more or fewer agents, machines, or storage areas without further training. It delivers more parts, better utilizes the machines, and causes fewer collisions. This makes SMAPPO a more suitable solution for real-world deployments where the production capacity of the industrial facility can increase or decrease depending on the market demands.

D. Curriculum with Short Initial Training

This experiment evaluated the model's adaptability with only 200 episodes of initial training (compared to 2000 in the main curriculum learning experiment V-B). As shown in Fig. 5, the model can achieve most of its learning (70% of its convergence performance) within this time. Then, similarly, more entities are introduced every 100 episodes to assess the ability with extremely short initial training.

The results in Fig. 9 and Table IV show that even with extremely short initial training, SMAPPO can learn to adapt faster to changes in the facility scale. SMAPPO greatly enhances parts deliveries and machine utilization while reducing collisions in large-scale setups.

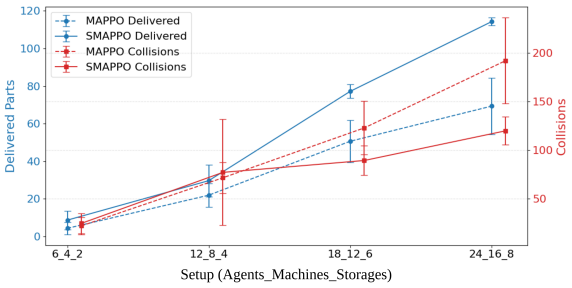

Fig. 9: Curriculum with short initial training parts delivery and collisions average and std as error bars.

TABLE IV: Curriculum with short initial training: machine utilization average and (std)

Setup	MAPPO	SMAPPO
6_4_2	0.17 (0.07)	**0.27 (0.11)**
12_8_4	0.33 (0.07)	**0.43 (0.09)**
18_12_6	0.48 (0.08)	**0.69 (0.03)**
24_16_8	0.48 (0.09)	**0.76 (0.01)**

Furthermore, additional experiments are provided in the supplementary materials, including training and model parameters, and more results and experiments to cover other aspects such as analysis of the learned attention, the effect of action and observation noise, maximum speed, and safety stops, in addition to targeted platforms and the practicality of the model.

VI. Conclusion

This work optimized an existing simulator to better support complex MARL scenarios with large numbers of agents and entities. Then, used it to build a multi-agent, multi-machine tending scenario and proposed a new scalable input-size invariant model (SMAPPO) to address that scenario in a decentralized manner. Moreover, it provided extensive experiments demonstrating the performance gain by SMAPPO in comparison to the state-of-the-art MAPPO, focusing on scalability to more agents and entities with full, minimum, and no retraining. In addition, we evaluated the model under extremely short initial training time.

We hope this work encourages other researchers to steer MARL research toward real-world scenarios, especially in the manufacturing sector. Future work could further explore real-world applicability concerns, such as the effect of communication issues. Additionally, the generalization of the systems could be further studied in different simulation layouts and the real world. Another direction could be to include raw material handling.

References

[1] E. Commission, D.-G. for Research, Innovation, M. Breque, L. De Nul, and A. Petridis, *Industry 5.0 – Towards a sustainable, human-centric and resilient European industry*. Publications Office of the European Union, 2021.

[2] T. Fan, P. Long, W. Liu, and J. Pan, "Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios," *arXiv preprint arXiv:1808.03841*, 2018.

[3] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, "The starcraft multi-agent challenge," *arXiv preprint arXiv:1902.04043*, 2019.

[4] K. Kurach, A. Raichuk, P. Stańczyk, M. Zając, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet *et al.*, "Google research football: A novel reinforcement learning environment," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 4501–4510.

[5] A. Agarwal, S. Kumar, and K. Sycara, "Learning transferable cooperative behavior in multi-agent teams," *arXiv preprint arXiv:1906.01202*, 2019.

[6] Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, and X. Wang, "Evolutionary population curriculum for scaling multi-agent reinforcement learning," *arXiv preprint arXiv:2003.10423*, 2020.

[7] G. Chen, S. Yao, J. Ma, L. Pan, Y. Chen, P. Xu, J. Ji, and X. Chen, "Distributed non-communicating multi-robot collision avoidance via map-based deep reinforcement learning," *Sensors*, vol. 20, no. 17, p. 4836, 2020.

[8] A. Iriondo, E. Lazkano, L. Susperregi, J. Urain, A. Fernandez, and J. Molina, "Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning," *Applied Sciences*, vol. 9, no. 2, p. 348, 2019.

[9] A. Iriondo, E. Lazkano, A. Ansuategi, A. Rivera, I. Lluvia, and C. Tubío, "Learning positioning policies for mobile manipulation operations with deep reinforcement learning," *International journal of machine learning and cybernetics*, vol. 14, no. 9, pp. 3003–3023, 2023.

[10] A. Agrawal, S. J. Won, T. Sharma, M. Deshpande, and C. McComb, "A multi-agent reinforcement learning framework for intelligent manufacturing with autonomous mobile robots," *Proceedings of the Design Society*, vol. 1, pp. 161–170, 2021.

[11] A. Abdalwhab, G. Beltrame, S. E. Kahou, and D. St-Onge, "Learning multi-agent multi-machine tending by mobile robots," *arXiv preprint arXiv:2408.16875*, 2024.

[12] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. M. Bayen, and Y. Wu, "The surprising effectiveness of mappo in cooperative," *Multi Agent Games*, 2021.

[13] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[14] X. Guo, D. Shi, J. Yu, and W. Fan, "Heterogeneous multi-agent reinforcement learning for zero-shot scalable collaboration," *arXiv preprint arXiv:2404.03869*, 2024.

[15] T. Hu, Z. Zhang, C. Zhu, G. Xu, Y. Wu, H. Wu, and Y. Liu, "Marf: Cooperative multi-agent path finding with reinforcement learning and frenet lattice in dynamic environments," in *2025 IEEE International Conference on Robotics and Automation (ICRA 2025)*. IEEE, 2025.

[16] M. Wen, J. Kuba, R. Lin, W. Zhang, Y. Wen, J. Wang, and Y. Yang, "Multi-agent reinforcement learning is a sequence modeling problem," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 509–16 521, 2022.

[17] W. Wang, T. Yang, Y. Liu, J. Hao, X. Hao, Y. Hu, Y. Chen, C. Fan, and Y. Gao, "From few to more: Large-scale dynamic multiagent curriculum learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 7293–7300.

[18] S. Thakar, P. Rajendran, A. M. Kabir, and S. K. Gupta, "Manipulator motion planning for part pickup and transport operations from a moving base," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 191–206, 2020.

[19] D. Chen, Z. Liu, and G. von Wichert, "Grasping on the move: A generic arm-base coordinated grasping pipeline for mobile manipulation," in *2013 European Conference on Mobile Robots*. IEEE, 2013, pp. 349–354.

[20] B. Burgess-Limerick, J. Haviland, C. Lehnert, and P. Corke, "Reactive base control for on-the-move mobile manipulation in dynamic environments," *IEEE Robotics and Automation Letters*, 2024.

[21] M. Bettini, R. Kortvelesy, J. Blumenkamp, and A. Prorok, "Vmas: A vectorized multi-agent simulator for collective robot learning," *The 16th International Symposium on Distributed Autonomous Robotic Systems*, 2022.