TOWARDS LARGE-SCALE MULTI-AGENT SHARED EXPERIENCE TASK SOLVING

Lex Janssens
s2989344

Koen Oppenhuis
s1692836

ABSTRACT

In this paper, we are looking into the framework Shared Experience Actor Critic (SEAC) (Christianos et al., 2020). In this multi-agent reinforcement learning (RL) framework, agents collectively learn via shared experiences within their loss function. This framework has been applied to few-agent problems, but is yet to be explored for many-agent problems. Many-agent settings often suffer from diverging policies due and complex interactions. We will be exploring many-agent tasks via validating the results made by Christianos et al. (2020), experimenting with many-agent tasks to look at attributes and impact of scaling, looking into transfer learning and curriculum learning compared to a baseline, and finishing it off by applying SEAC to Electrical Vehicle charging task. We show that claims made within Christianos et al. (2020) do not necessarily apply to the properties of SEAC as described. Furthermore, we show that coordination is complex for many-agent environments, and that transfer learning and curriculum learning show decay in performance when scaled up to settings the configuration was not trained in.

1 INTRODUCTION

Exploring environment dynamics and the joint action space among agents is a critical aspect of multi-agent reinforcement learning (MARL). This is a difficult problem, because of the non-stationarity between agents, and the the exponentially growing action space in comparison to the number of agents. This problem stands out even more in sparse reward environments.
A method for more efficient MARL exploration would be to share experiences between agents. In sparse reward environments, when one agent finds a reward, it would be beneficial for the other agents to also know how to get this reward. This is possible by sharing this experience with the other agents. Christianos et al. (2020) proposes this framework in the form of Shared Experience Actor Critic (SEAC). SEAC updates an agent's actor and critic parameters by merging the gradients derived from the own agent's experience together with the weighted gradients derived from the other agent's experiences. It is claimed that SEAC learns substantially faster and achieves higher final rewards in several sparse-reward multi-agent environments, compared to multiple MARL baselines.
To verify these claims, we attempted to reproduce some experiments done in this paper. We noticed the small scale these experiments were performed on and were driven to find out whether this is applicable to other problems, especially problems where many agents have to solve a task cooperatively. For this, we have experimented with the attributes of SEAC when scaled to coordinate on a larger scale. This includes assessing the impact of different ways of scaling the number of agents of a task using this framework, including transfer learning and curriculum learning. With this knowledge, we aim to apply it to a simulation of Electrical Vehicle (EV) charging (Yeh et al., 2023).

In this paper, we validate the results made by Christianos et al. (2020). Therefore, we perform six experiments, five of which were also performed in that paper. Furthermore, we will be looking into attributes and impact of scaling the number of agents using the framework SEAC. We do this by comparing policies and trajectories of trained agents, interchange agents within configurations and scaling the number of agents within a configuration via transfer learning and curriculum learning to be compared against a baseline. Lastly, we will be applying SEAC to EVCharging, using the information obtained by the previous experiments.

2 BACKGROUND/PRELIMINARIES

We define a Partially Observable Markov Decision Process (POMDP) for a multi-agent RL setting with N agents as the tuple $(N, \mathcal{S}, \{O^i\}_{i\in[N]}, \{A^i\}_{i\in[N]}, \Omega, \mathcal{P}, \{\mathcal{R}^i\}_{i\in[N]})$, where $\mathcal{S}$ defines the state space, $\mathcal{O} = O^1 \times \cdots \times O^N$ defines the joint observation space, $\mathcal{A} = A^1 \times \cdots \times A^N$ defines the joint action space, $\Omega : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{O})$ as the observation function returning for each agent i an individual observation $o^i \in O^i$, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ the transition function, $\mathcal{R}^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the reward function for agent i. The goal is to find policies $\Pi^* = (\pi_1^*, \ldots, \pi_N^*)$, where each pi_i produces the maximized discounted return $G^i = \sum_{t=1}^{T} \gamma^t r_t^i$ w.r.t. all other policies. Christianos et al. (2020) formally defines this as $\forall i : \pi_i \in \arg\max_{\pi_i'} \mathbb{E}[G_i|\pi_i', \Pi\backslash\{\pi_i\}]$. Here, t is the current time step, T is the total number of timesteps of an episode, and γ is the discount factor. It is assumed that the observation spaces and action spaces of each agent are identical, i.e. $O = O^1 = \cdots = O^N$ and $A = A^1 = \cdots = A^N$. However, each independent agent has a separate reward function.

An Actor-Critic (AC) model is part of the class of Policy Gradient (PG) algorithms, consisting of two parts: an acting policy within the environment, and a variance-reducing value function. An PG algorithm aims to directly find policy π_ϕ by updating ϕ using the negative gradient of a loss function to be optimized. In this case, ϕ are the parameters defining a neural network. Furthermore, the value function V_π is a parameterized neural network via parameters θ, and is updated in the process to enhance the Monte Carlo estimates made by this network. Given agent i, a loss function with policy parameters ϕ^i and value function parameters θ^i, the loss functions to be minimized are respectively defined as

$$L(\phi_i) = -\log \pi(a_t^i|o_t^i, \phi_i)(r_t^i + \gamma V(o_{t+1}^i; \theta_i) - V(o_t^i; \theta_i)) \tag{1}$$

and

$$L(\theta_i) = ||V(o_t^i; \theta_i) - y_i||^2 \text{ with } y_i = r_t^i + \gamma V(o_{t+1}^i; \theta_i) \tag{2}$$

For checking the difference between two RL policies, we can make use of statistical methods of comparing distributions. In this paper, we will be dealing with a discrete sample space of size n. A function $p : X \to [0, 1]$, parameterized by a neural network, assigns probabilities to each element in the sample space while cohering to the constraint that $\sum_{i=1}^{n} p(x_i) = 1$, where $X = \{x_1, \ldots, x_n\}$. To compare two distributions p and q where the sample space is of equal size, we will be using the Kullback-Leibler Divergence (Kullback & Leibler, 1951):

$$KL(p||q) = \sum_{i=1}^{n} p(x_i) \log \frac{p(x_i)}{q(x_i)} \tag{3}$$

This formula tells us how much p diverges from q represented by a value $KL(p||q) \in [0, \infty)$. This formula is assymetric, i.e. $KL(p||q) \not\equiv KL(q||p)$. Another (symmetric) formula would be Jensen-Shannon divergence, which uses a mixture distribution of p and q in its calculation (Lin, 1991).

3 RELATED WORK

As Papoudakis et al. (2019) has pointed out, MARL suffers from non-stationarity due to simultaneously learning agents with a joint action space. This problem can be solved in multiple ways, such as working with *centralized training and decentralized execution* (CTDE) (Kraemer & Banerjee, 2016; Lowe et al., 2017; Oliehoek et al., 2008; Rashid et al., 2020), where agents have access to all observations, as such observations of other agents, for training purposes. At execution time, only the local observation is used to determine the next action. Kraemer & Banerjee (2016) has shown that this stabilizes the local policies of agents, regardless of divergence of other agents' policies. SEAC uses CTDE not only to learn from own observations, but also learn from other agents' (weighted) observations.

This also differs from an *agent-teaching-agent* setting, where a learner agent gets evaluated and improved via a teacher agent, pointed out via instructions (Da Silva et al., 2019). Other works incorporate this concept via knowledge exchange (da Silva et al., 2017; 2018), though SEAC shares trajectories such that agents may use this information in their way, which may differ in importance among other agents.

Since the agents are learning simultaneously in the SEAC framework, the concept of *learning from demonstration* is neglected. This method let agents learn a policy via an example, such as a human expert (Taylor et al., 2011), trajectories (Schaal, 1996), or a dataset (Ho & Ermon, 2016; Zimmer et al., 2014). Additionally, the tasks are not too expensive such that sample efficiency is a must.

Single-agent RL may make use of *distributed RL*: using multiple agents to learn a single agent, via efficient data generation and decorrelation (A3C) (Mnih et al., 2016), or distribution of data over various learners but optimizing on a single learner (IMPALA and SEED RL) (Espeholt et al., 2018; 2020). This speeds up learning a single agent, but has only little to do with simultaneous learning in a MARL setting. Using multiple sets of agents (such as *population-play*) may improve multiple agents simultaneously via improved exploration and coordination, it is highly expensive (Wang et al., 2018; Leibo et al., 2018; Jaderberg et al., 2018; Long et al., 2020).

The problem of upscaling the number of agents in a multi-agent system (MAS) is a known problem. When scaling up the number of agents in a MAS, not only computational complexities arise, but also communication constraints and coordination difficulties. Liu et al. (2024) have classified three classes of solutions. *Knowledge sharing* is a collection of methods that allow agents to benefit from the knowledge of other agents. Parameters sharing (Gupta et al., 2017) helps speed up training in MARL settings and makes it essentially scale to perform well on many-agent tasks. This approach however produces agents with identical policies, which may not be applicable in certain tasks. Transfer learning considers knowledge from a previously learned task to be reused in the current task (Shao et al., 2018). While this may avoid a cold-start problem, this does not scale with the number of agents, since the tasks between few-agent and many-agent increasingly differ. Lastly, *curriculum learning* aims to improve agents' performance by gradually scaling up the task's difficulty. By learning basic skills such as navigation in a simple task, learning coordination among agents may be introduced later, matching the potential an agent may have at a certain stage and benefit in the long-term (Narvekar et al., 2016; Elman, 1993). Various studies (Long et al., 2020; Shao et al., 2018; hu, 2020; Wang et al., 2019) show faster convergence speed and better performance when using curriculum learning in many-agent settings. We will apply curriculum learning to scale up the number of agents in SEAC, such that navigation and basic coordination are learned first and advanced coordination is realized later.

Sample efficiency focuses on data-efficiency, dealing primarily with model accuracy and approaching large computational complexity issues. Inverse RL (Yu et al., 2019; Song et al., 2018) and attention mechanisms (Jeon et al., 2020) try to solve these problems, but require deriving the reward function via an expert. SEAC neglects the added complexity of an attention mechanism and does not use the knowledge of an expert, but instead uses shared knowledge between agents to be sample efficient. *Simplification* focuses on reducing the computational complexity by assuming interactions between all agents are considered redundant, and that large-N agent environments have much more complex relationships between agents, only a dynamically select few agents have interactions between them (Iqbal & Sha, 2019). However, considering we are using sparse-reward environments, information between agents is more valuable and may introduce oversimplification. SEAC accepts this problem as it is for now, but it may be looked into to reduce interactions over training time.

4 Methodology

4.1 Algorithmic description

In SEAC, every agent generates one on-policy trajectory. In general, the agent uses experiences from its own sampled trajectory, and updates its network with equation 1. However, in SEAC the agent also uses the trajectories of the other agents. This is considered off-policy data because these agents have a different policy than the agent itself. To correct the off-policy samples to the own policy, importance sampling is used. If we extend the policy loss in equation 1 with the off-policy data from other agents you get equation 4.

$$\begin{aligned} L(\phi_i) = &- \log \pi(a_t^i|o_t^i, \phi_i)(r_t^i + \gamma V(o_{t+1}^i;\theta_i) - V(o_t^i;\theta_i)) \\ &- \lambda \sum_{k\neq i} \frac{\pi(a_t^k|o_t^k;\phi_i)}{\pi(a_t^k|o_t^k;\phi_k)} \log \pi(a_t^k|o_t^k;\phi_i)(r_t^k + \gamma V(o_{t+1}^k;\theta_i) - V(o_t^k;\theta_i)) \end{aligned} \tag{4}$$

In this equation, the agent itself is denoted with i, and the other agents are denoted with k. We can extend the value loss from equation 2 similarly with the off-policy data from other agents to yield equation 5.

$$\begin{aligned} L(\theta_i) &= ||V(o_t^i;\theta_i^i) - y_i||^2 + \lambda \sum_{k \neq i} \frac{\pi(a_t^k|o_t^k;\phi_i)}{\pi(a_t^k|o_t^k;\phi_k)} ||V(o_t^i;\theta_i) - y_k^i||^2 \\ y_k^i &= r_t^k + \gamma V(o_{t+1}^k;\theta_i) \end{aligned} \tag{5}$$

The hyperparameter λ weights the experiences of the other agents. For this research, this value is kept at 1. With the use of equations 4 and 5, each agent is now trained on its on-policy data, and the off-policy data from the other agents. The pseudocode for SEAC is found in Algorithm 1.

Algorithm 1 Shared Experience Actor-Critic Framework Christianos et al. (2020)

```
1: for timestep $t = 1\ldots$ do
2:     Observe $o_t^1, \ldots, o_t^N$
3:     Sample actions $a_t^1, \ldots, a_t^N$ from $P(o_t^1;\phi_1), \ldots, P(o_t^N;\phi_N)$
4:     Execute actions and observe $r_t^1, \ldots, r_t^N$ and $o_{t+1}^1, \ldots, o_{t+1}^N$
5:     for agent $i = 1, \ldots, N$ do
6:         Perform gradient step on $\phi_i$ by minimizing Equation 4
7:         Perform gradient step on $\theta_i$ by minimizing Equation 5
8:     end for
9: end for
```

Upscaling is done via zero-shot transfer learning and curriculum learning. Via transfer learning, we let agents train for a set amount of timesteps in a configuration of N agents. We will evaluate these agents on a setting where the number of agents are larger to see whether retraining is necessary. In curriculum learning, we will use trained agents to be trained in a setting where N is larger than the original trained setting, for a set amount of time.

To compare two policies, we will use an adaptation of the KL-divergence metric. Rewriting the KL-divergence to compare two policies given an observation o, we yield $KL(\pi_i||\pi_j|o) = \sum_{a\in\mathcal{A}} \pi_i(a|o;\phi^j) \log \frac{\pi_i(a|o;\phi^i)}{\pi_j(a|o;\phi^j)}$. To compare two policies, an average of $\forall o \in O$ should be considered. Using the law of large numbers, we can approximate this by taking a large enough set of observations in O, such that

$$\mathbb{E}[KL(\pi_i||\pi_j)] \approx AKL(\pi_i||\pi_j|O) = \frac{1}{|O|} \sum_{k=1}^{|O|} \sum_{a\in\mathcal{A}} \pi_i(a|o_k;\phi^i) \log \frac{\pi_i(a|o_k;\phi^i)}{\pi_j(a|o_k;\phi^j)}, o_k \in O \tag{6}$$

4.2 Experimental design

We evaluate SEAC on three different environments. Level Based Foraging (LBF) and Robotic Warehouse (RWARE) (Papoudakis et al., 2021) come from the paper by Christianos et al. (2020), the other environment is part of SustainGym (Yeh et al., 2023) [1].

LBF and RWARE are sparse task multi-agent environments. The task in LBF is to cooperatively collect rewards in the environment. A reward r may only be collected if the agents neighboring this reward have a collective level equal or higher than r. This obligates agents to navigate to the same reward and coordinate whether one agent may be applicable for a certain reward. In a forced cooperative setting, all agents need to neighbor some reward to get it. RWARE instead focuses on a collective task via individual contributions. Navigating boxes throughout a warehouse may yield rewards if these are delivered (and eventually brought back). Both environments may differ in size and or the number of rewards available for adjusted difficulty. See appendix D for more details.

The multi-agent EVCharging environment is part of the SustainGym environments Yeh et al. (2023), which aims to bring RL to sustainability problems. In the multi-agent EVCharging environment, each agent controls one EV charging station. Each agent can decide how much charge it delivers at

[1] Results and trained agents may be found at `https://github.com/Lexpj/ARL-SEAC`

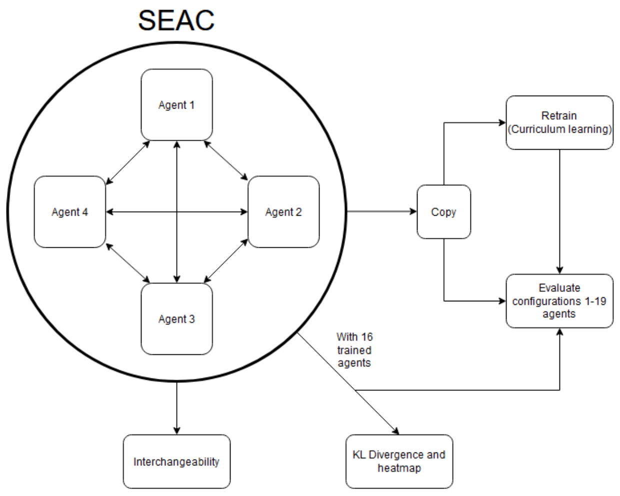

SEAC
Agent 1
Agent 2
Agent 3
Agent 4
Copy
Retrain (Curriculum learning)
Evaluate configurations 1-19 agents
With 16 trained agents
Interchangeability
KL Divergence and heatmap

Figure 1: Overview of all experiments done considering upscaling

any time of day. There are limitations on the energy grid, so not all stations can use the maximum charging rates simultaniously. The goal is to maximize the charge delivery while minimizing the carbon costs.

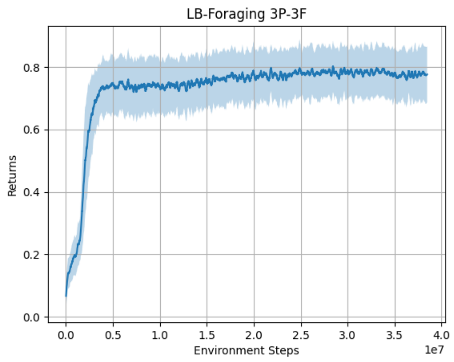
4.2.1 Reproduction

With reproduction, we aim to validate the claims made in Christianos et al. (2020). Therefore, we have run six different experiments collectively on LBF and RWARE. To ensure that the experiments were similar to the experiments in the original paper, the same code base was used[2]. These experiments use the same hyperparameter configuration (which is also included in appendix A), the same amount of total timesteps T, and performed over 5 repetitions.

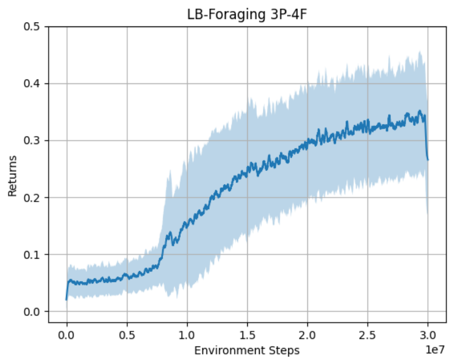
4.2.2 Upscaling

In terms of upscaling experiments, we have looked at four attributes. First, Christianos et al. (2020) claims that trained policies are similar but not identical. Our experiment involves training $N = 16$ agents for 7 days walltime and observing their trajectories along with KL divergence between the policies to see whether this still holds for large N. KL divergence considers critical states: all the collections of observations $\{o^i_{t-1}, o^i_t, o^i_{t+1}\}$ an agent comes across in its trajectory where arg max$\pi(o^i_t; \phi_i)$ is the action of picking up or dropping a box. Second, we want to measure the importance of coordination within this environment, as well as give us insight into the impact of using a different configuration of agents than it has been trained on. Using $N = 4$ trained agents, and performing 100 evaluations, we interchange agents to see whether this influences the performance, positively or negatively. The different configurations are denoted by a tag $a_1a_2a_3a_4$, where each a is the count of agents of that ID. For example, 2002 denotes that 2 trained agents 1 are used and 2 trained agents 4. The default configuration 1111 is shown in green, and the configurations where only one agent ID is used are shown in red. Third, we compare different ways of scaling up the number of agents for a task. Hereby we will look at the differences between transfer learning and curriculum learning. For transfer learning, four sets of four trained agents are evaluated. Similarly, in curriculum learning, four sets of four trained agents are retrained (7 days walltime) as a configuration of 16 agents and evaluated. These results are compared against a baseline of 16 trained agents, both in a normal setting and a sparse setting. The results are gathered via the average of 100 accumulated trajectory returns of the environment, divided over the number of agents. The configurations are all trained in the same environment, regardless of the number of agents.

[2]This code base is publicly available on `https://github.com/semitable/seac`



(a) LBF ((10×10, 3 agents, 3 rewards)



(b) LBF ((15×15), 3 agents, 4 rewards)

Figure 2: (a) and (b) show the reward per episode during training of SEAC on to their respective environments. The reward is equal to the percentage of the levels of the rewards that is collected per episode. In (a), SEAC converges to a higher reward than in (b). Because of the smaller grid size and the less reward that can be collected.

4.2.3 Electrical Vehicle Charging

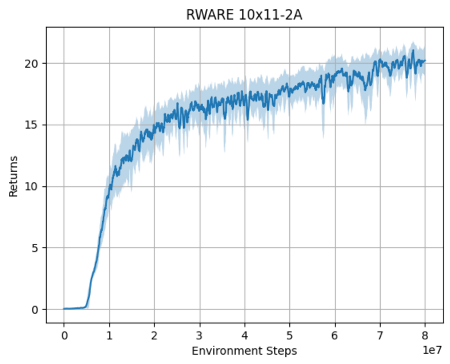
Lastly, we use SEAC on the EVCharging problem by SustainGym (Yeh et al., 2023) and compare it to the baselines provided in this work. To train SEAC agents on the EVCHarging problem, data is sampled from a Gaussian Mixed model, fit to historical data. This data is gathered from the Caltech EV charging network in the summer of 2019 (Lee et al., 2019). The training is done for 250.000 timesteps. After training the agents are tested on real historical data, gathered in the summer of 2021.

5 Results

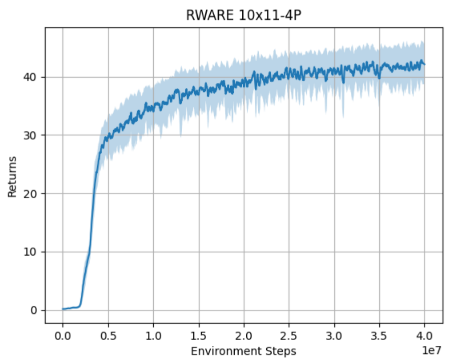
We divide up the results in sections *reproduction* and *upscaling*, covering the research questions for each part respectively. The research question for SEAC applied on EVCharging is dealt with in appendix B.

5.1 Reproduction

For the verification of the claims found in the paper by Christianos et al. (2020), six experiments were reproduced. First, we will look at the two non-cooperative experiments of the LBF environment.
In figure 2a, the reward curve is shown for SEAC in an LBF environment with a grid size of 10×10, $N = 3$ agents and 3 rewards. Comparable to the results obtained by Christianos et al. (2020), SEAC starts finding and collecting rewards rather quickly and converges to an average of just below 0.8.
In figure 2b, the reward curve is shown for SEAC in an LBF environment with a grid size of 15×15, $N = 3$ agents and 4 rewards. Because of the larger environment size, this task is sparser than the previous task. This causes different runs to be inconsistent as to when the agents start learning, which is only due when the agents find the first reward. Because of the grid size and the amount of rewards, it is often impossible to collect them all, and therefore the reward converges to just above 0.3. This is in line with Christianos et al. (2020). For the RWARE environment, SEAC was tested on the tiny size with 2 agents and with 4 agents. In figure 3, the results of these experiments are shown. Both results are similar to the results found in the paper by Christianos et al. (2020). This indicates that there is less variance in finding a reward compared to the LBF experiments.
If you compare the difference in the start point of learning between both experiments, you can see that with 4 agents SEAC starts learning twice as fast compared to 2 agents. This can again be explained by the finding of a reward for the first time. With 4 agents this is twice as likely compared to 2 agents. To verify if SEAC works well in cooperative environments, it was also tested on LBF in a cooperative setting. In figure 4a, the results of the 5 different training runs are shown. The runs would either converge at 0.6 rewards per episode or 0.3 rewards per episode. This indicates that the

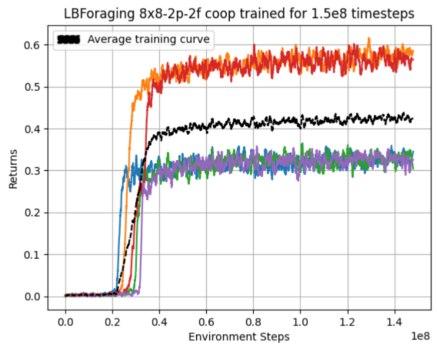


(a) RWARE (tiny, 2 agents, 2 boxes)

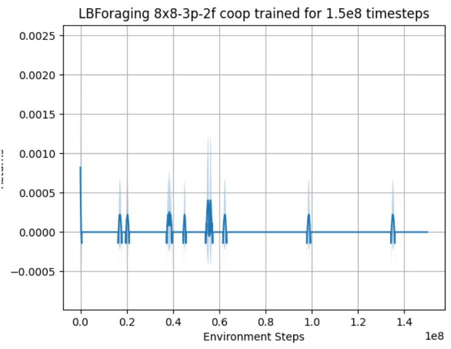


(b) RWARE (tiny, 4 agents, 4 boxes)

Figure 3: (a) and (b) show the reward per episode during training of SEAC on their respective environments. In the RWARE environment, it is difficult to collect the first positive reward. After this, both (a) and (b) show that SEAC is efficient in sharing this experience and that the total reward per episode increases quickly. Because (b) has twice as many agents, the initial learning starts at half the environment steps as in (a).



(a) LBF-coop ((8×8, 2 agents, 2 rewards)



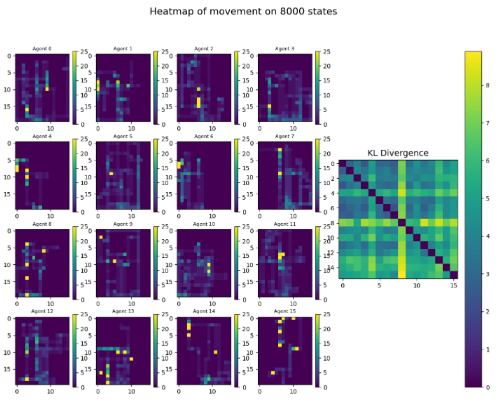
(b) LBF-coop ((8×8, 3 agents, 2 rewards)

Figure 4: (a) shows 5 different training runs on the cooperative setting of the LBF environment with 2 agents and 2 rewards. Because of the convergence of 2 separate values, we can say that SEAC did not always find the optimal policy for this environment. (b) shows the reward during training in the cooperative setting of the LBF environment with 3 agents and 2 rewards. The agents failed to cooperate and collect rewards together.

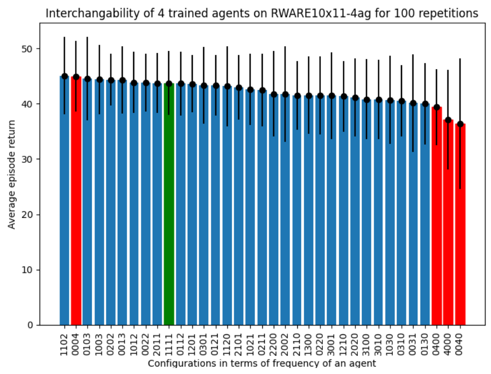
starting seed for this environment is important. Christianos et al. (2020) has shown no significant differences between runs, possibly omitting this artifact.
In figure 4b the rewards for LBF on an 8×8 grid with 3 agents and 2 rewards is shown. In 150 million timesteps SEAC did not learn to achieve a higher return value. This suggests difficulty in the cooperation between different agents.
SEAC does not help with this kind of cooperation, because SEAC helps mostly with faster exploration. In these cooperative environments, both agents get a reward at the same time, and therefore they need less help from the experiences of the other agent.

5.2 Upscaling

In the first experiment, seen in figure 5a, we have trained 16 agents on RWARE of size 16×20. We then proceeded to expose the similarities between agents, via KL-divergence of critical states and a heatmap tracing each agent's trajectory. What we observe is that some policies are highlighted in the KL-divergence plot, meaning that these differ from other policies. These are agents (0-indexed) 4, 8, 13, 14 and 15. We can confirm via the heatmaps that these agents are not contributing as much to the task and mainly get stuck at specific parts of the environment. Looking at the remaining agents, we can see that some coordination has been integrated. For example, agents 6, 3, and 7 seem to divide the task to focus mainly on the left, middle, and right sides of the environment respectively.



(a) Similarity



(b) Interchangeability

Figure 5: Results of (a) show the similarity between policies for a configuration where 16 agents are trained for 7 days wall time, in terms of KL-divergence of 8000 critical states and trajectory in the environment. It shows that even though most policies benefit from each other and some coordination is integrated with learning, some policies are behind in contributing to the environment task. Results of (b) show the interchangeability when using 4 trained agents, by using different configurations where some agents may be left out and some may be included in the configuration multiple times, evaluated for 100 runs. It shows that some agents are beneficial to include within a configuration, regardless of coordination between other agents or the original configuration on which it has been trained.

In figure 5b, we have done a study on the interchangeability of agents. As you can see, some configurations outperform the configuration the agents were trained on. In 3 out of 4 cases, using only one trained agent ID results in poor performance. However, this shouldn't necessarily be the case, as seen in this experiment. We can also correlate the performance to the number of agents used per ID in each configuration, showing that agent ID 4 seems to show an overall performance and is present in most top configurations, while other agent IDs lack.

Figure 6 shows the results of both using zero-shot transfer learning and curriculum learning in different settings, using various amounts of agents and structurally built up. Appendix C shows this same experiment, but the agent configuration is chosen randomly. We compare transfer-learning (red) and curriculum learning (orange) to baselines normal (green) and sparser (blue) configurations. We can see that performance is maximal around $N = 6$, but decays in performance if fewer or more agents are picked. An explanation could be that the learned coordination has the potential to handle up to 6 agents fairly well, but cannot deal with more interactions since this has not been properly learned. For $N = 16$ agent settings, we see that similar behavior occurs, but around the $N = 19$ mark. We see that lower values for N show a decrease in performance, which could imply that coordination for large N does not translate to small N in the same way, resulting in two completely different forms of coordination. Lastly, in orange, we see that a combination of two different coordinating policies is integrated within the agents, seeing a peak performance at $N = 6$ and $N = 19$. This result, compared to zero-shot transfer learning, shows that coordination at some number of agents must be learned at some point to perform well. Downscaling the number of agents does not show a decrease in performance, but upscaling in this way would.

6 DISCUSSION

In this paper, we addressed the problem of scaling SEAC (Christianos et al., 2020) towards many-agent problems, via validating the original results, looking at the attributes and methods of scaling the number of agents, and applying it to a many-agent problem. We see that some of the results claimed by the paper are off. Furthermore, we see that some claims made for SEAC do not neces-

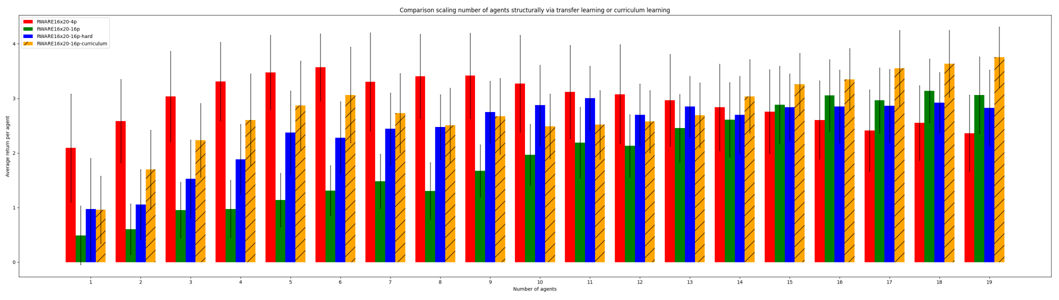


Figure 6: Comparison between the average return per agent averaged over 100 evaluations using different N. Configurations include 4 agents along with zero-shot transfer learning (red), 16 trained agents (green), 16 trained agents in a sparse environment (blue), and 4 trained agents retrained with 16 agents as a way of curriculum learning (orange). We can see that performance peaks just above the configurations of N for each setting that it has been trained. Downscaling is possible, but upscaling requires retraining (most efficiently) using curriculum learning.

sarily apply to the properties of SEAC, as seen in the cooperative LBF experiments. Furthermore, we have seen that scaling the number of agents does not only increase the training time but also the complexity of coordination between agents. Agents do however tend to coordinate to a certain extent, also for many-agent settings. Interchangeing agents may outperform the configuration it was originally trained on, but is mainly due to one well-performing agent responsible for the collective cumulated obtained reward. Using agent configurations on environments where more-agent configurations are expected, seems to underperform due to a lack of coordination. Transfer learning shows that, although easy in complexity, underperforms when it comes to many-agent settings. Curriculum learning shows that both few-agent settings as well as many-agent settings maintain learned coordination, but face similar problems when scaling to larger problems.

We see that the collection of problems Liu et al. (2024) stated when upscaling the number of agents for tasks remain for SEAC. For one, Shao et al. (2018) stated that tasks seem to differ between few-agent and many-agent configurations. We can conclude that this is true since complex coordination becomes a major task in itself.

One significant hurdle was our decision to implement SEAC from scratch rather than using the original code base. This choice was made to gain a deeper understanding of the algorithm's inner workings. However, the complexity of SEAC and the debugging process took more time than anticipated. Consequently, we needed to fall back to the original code base. Despite this, the experience provided us with insights into the inner workings of SEAC and helped with understand the original code base. Another limitation was our difficulty in scaling SEAC to the EVCharging environment. Because of memory constraints, we couldn't get SEAC to train on this environment. This would be something to look into for future work.

There are many other directions for future work. First, applying SEAC to a real-life problem would provide a crucial test of its practical viability. Additionally, exploring SEAC's performance in a dense reward environment would be interesting. SEAC is good in exploration, and it would be interesting to see if the sharing of experience that benefit it in sparse reward environments also translate to more dense reward environments. Understanding this could help the algorithm and expand its use cases.

REFERENCES

Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. *Neural Information Processing Systems*, 33:10707–10717, 12 2020. URL `https://proceedings.neurips.cc/paper/2020/file/7967cc8e3ab559e68cc944c44b1cf3e8-Paper.pdf`.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

Felipe Leno da Silva, Ruben Glatt, and Anna Helena Reali Costa. Simultaneously learning and advising in multiagent reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pp. 1100–1108, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

Felipe Leno da Silva, Matthew E. Taylor, and Anna Helena Reali Costa. Autonomously reusing knowledge in multiagent reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2018. URL https://api.semanticscholar.org/CorpusID:51604728.

Felipe Leno Da Silva, Garrett Warnell, Anna Helena Reali Costa, and Peter Stone. Agents teaching agents: a survey on inter-agent transfer learning. *Autonomous Agents and Multi-Agent Systems*, 34(1), 12 2019. doi: 10.1007/s10458-019-09430-0. URL https://doi.org/10.1007/s10458-019-09430-0.

Jeffrey L. Elman. Learning and development in neural networks: the importance of starting small. *Cognition*, 48(1):71–99, 1993. ISSN 0010-0277. doi: https://doi.org/10.1016/0010-0277(93)90058-4. URL https://www.sciencedirect.com/science/article/pii/0010027793900584.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. *CoRR*, abs/1802.01561, 2018. URL http://arxiv.org/abs/1802.01561.

Lasse Espeholt, Raphaël Marinier, Piotr Michal Stanczyk, Ke Wang, and Marcin Michalski. Seed rl: Scalable and efficient deep-rl with accelerated central inference. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1910.06591.

Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In Gita Sukthankar and Juan A. Rodriguez-Aguilar (eds.), *Autonomous Agents and Multiagent Systems*, pp. 66–83, Cham, 2017. Springer International Publishing. ISBN 978-3-319-71682-4.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016. URL http://arxiv.org/abs/1606.03476.

Chunyang hu. A confrontation decision-making method with deep reinforcement learning and knowledge transfer for multi-agent system. *Symmetry*, 12:631, 04 2020. doi: 10.3390/sym12040631.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2961–2970. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/iqbal19a.html.

Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio García Castañeda, Charlie Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364:859 – 865, 2018. URL https://api.semanticscholar.org/CorpusID:49561741.

Wonseok Jeon, Paul Barde, Derek Nowrouzezahrai, and Joelle Pineau. Scalable multi-agent inverse reinforcement learning via actor-attention-critic. *CoRR*, abs/2002.10525, 2020. URL https://arxiv.org/abs/2002.10525.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2016.01.031. URL https://www.sciencedirect.com/science/article/pii/S0925231216000783.

S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951. doi: 10.1214/aoms/1177729694. URL `https://doi.org/10.1214/aoms/1177729694`.

Zachary J. Lee, Tongxin Li, and Steven H. Low. ACN-Data: Analysis and Applications of an Open EV Charging Dataset. In *Proceedings of the Tenth International Conference on Future Energy Systems*, e-Energy '19, June 2019.

Joel Z. Leibo, Julien Pérolat, Edward Hughes, Steven Wheelwright, Adam H. Marblestone, Edgar A. Duéñez-Guzmán, Peter Sunehag, Iain Dunning, and Thore Graepel. Malthusian reinforcement learning. *CoRR*, abs/1812.07019, 2018. URL `http://arxiv.org/abs/1812.07019`.

J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. doi: 10.1109/18.61115.

Dingbang Liu, Fenghui Ren, Jun Yan, Guoxin Su, Wen Gu, and Shohei Kato. Scaling up multi-agent reinforcement learning: An extensive survey on scalability issues. *IEEE Access*, 12:94610–94631, 1 2024. doi: 10.1109/access.2024.3410318. URL `https://doi.org/10.1109/access.2024.3410318`.

Qian Long, Zihan Zhou, Abhinav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. Evolutionary population curriculum for scaling multi-agent reinforcement learning. *CoRR*, abs/2003.10423, 2020. URL `https://arxiv.org/abs/2003.10423`.

Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf`.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/mniha16.html`.

Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. Source task creation for curriculum learning. 05 2016.

F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, May 2008. ISSN 1076-9757. doi: 10.1613/jair.2447. URL `http://dx.doi.org/10.1613/jair.2447`.

Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V. Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *CoRR*, abs/1906.04737, 2019. URL `http://arxiv.org/abs/1906.04737`.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.*, 21(1), January 2020. ISSN 1532-4435.

Stefan Schaal. Learning from demonstration. In M.C. Mozer, M. Jordan, and T. Petsche (eds.), *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. URL `https://proceedings.neurips.cc/paper_files/paper/1996/file/68d13cf26c4b4f4f932e3eff990093ba-Paper.pdf`.

Kun Shao, Yuanheng Zhu, and Dongbin Zhao. Starcraft micromanagement with reinforcement learning and curriculum transfer learning. *CoRR*, abs/1804.00810, 2018. URL `http://arxiv.org/abs/1804.00810`.

Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. *CoRR*, abs/1807.09936, 2018. URL `http://arxiv.org/abs/1807.09936`.

Matthew Taylor, Halit Suay, and Sonia Chernova. Integrating reinforcement learning with human demonstrations of varying ability. volume 1, pp. 617–624, 01 2011.

Jane X. Wang, Edward Hughes, Chrisantha Fernando, Wojciech M. Czarnecki, Edgar A. Duéñez-Guzmán, and Joel Z. Leibo. Evolving intrinsic motivations for altruistic behavior. *CoRR*, abs/1811.05931, 2018. URL `http://arxiv.org/abs/1811.05931`.

Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. From few to more: Large-scale dynamic multiagent curriculum learning. *CoRR*, abs/1909.02790, 2019. URL `http://arxiv.org/abs/1909.02790`.

Christopher Yeh, Victor Li, Rajeev Datta, Julio Arroyo, Nicolas Christianson, Chi Zhang, Yize Chen, Mehdi Hosseini, Azarang Golmohammadi, Yuanyuan Shi, Yisong Yue, and Adam Wierman. SustainGym: Reinforcement learning environments for sustainable energy systems. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, New Orleans, LA, USA, 12 2023. URL `https://openreview.net/forum?id=vZ9tA3o3hr`.

Lantao Yu, Jiaming Song, and Stefano Ermon. Multi-agent adversarial inverse reinforcement learning. *CoRR*, abs/1907.13220, 2019. URL `http://arxiv.org/abs/1907.13220`.

Matthieu Zimmer, Paolo Viappiani, and Paul Weng. Teacher-student framework: A reinforcement learning approach. 05 2014.

A Experimental details

The base model of the SEAC model is an A2C (Mnih et al., 2016) model, with 2 hidden layers both of 64 nodes, and 2 output heads: a policy and value head respectively. An n-step of 5 is used. In terms of hyperparameters, we are using the same settings as Christianos et al. (2020). This includes a learning rate $\alpha = 3e^{-4}$, Adam epsilon of 0.001, discount factor $\gamma = 0.99$, entropy coefficient of 0.01, value loss coefficient of 0.5, and gradient clipping at 0.5. Although GAE support is included within the code, it is not used during these experiments. Similarly, support of recursion within the neural network via a GRU (Chung et al., 2014) is available, but not used. The weight λ of the off-policy loss via the sum of combined losses of other agents remains at $\lambda = 1.0$, considering this attribute is insensitive when λ approaches 1.0 (Christianos et al., 2020).

The code has been run using the ALICE HPC cluster of Leiden University. Although inefficient, we have been using a single core of an Intel Xeon Gold 6126 2.6GHz 12-Core. This is due to problems logging problems when using multiple cores, overwriting previous results, or accessing the same file simultaneously, causing the run to stop. The environment was parallelized to run 4 simultaneous processes.

Unlike the original requirements of Christianos et al. (2020), we have maintained the list of packages within our code base to let it run on Python 3.8 and PyTorch 1.8.1.

B Discussion/Conclusion RQ3

The adaptation from the original code base of SEAC proved to be more difficult than anticipated. The dependencies of the original code base were outdated, and were not compatible with the dependencies of the SustainGym environments. After fixing these problems we tried running experiments on ALICE. However, this caused a different set of problems. The original code base was not build for using a large amount of agents. For EVCharging 54 agents are needed, and this caused memory

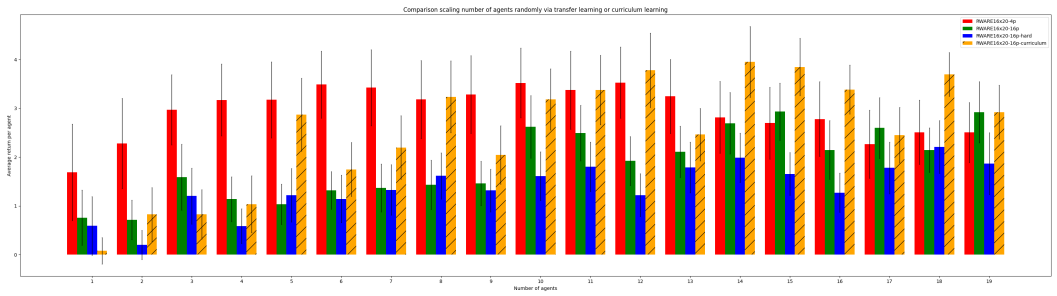


Figure 7: Comparison between the average return per agent averaged over 100 evaluations using different N. Configurations include 4 agents along with zero-shot transfer learning (red), 16 trained agents (green), 16 trained agents in a sparse environment (blue), and 4 trained agents retrained with 16 agents as a way of curriculum learning (orange). Performance is comparable to figure 6, though due to the randomness of the configurations, fluctuates more.

issues, before we could do any real measurements.
For further research, it would be interesting to create a clean code base for SEAC, and see how the algorithm would perform in problems similar to EVCharging. In the original paper about SEAC the algorithm is only tested on sparse, small-N environments. In dense, large-N environments it is not yet know how SEAC performs.

C Comparison zero-shot transfer learning and curriculum learning

We have run the same experiment in terms of comparison between different configurations using a variable N. In the paper, we did this structurally, only taking the first modulo N agents and for larger N, the first in line. Additionally, we have experimented by taking N random agents. We yield the results seen in figure 7. The results show similar results as figure 6, though due to the randomness of how the configurations are chosen, the results fluctuate more between runs, causing a more unstable average performance.

D Environmental details

LBF is a cooperative multi-agent environment in which multiple agents are trying to find and collect food. Every agent and every food has its own level. If the level of an agent is lower than that of the food, it has to cooperate with an other agent to get to the required level for collecting the food. There are 2 settings for this environment, in the normal setting some food has a lower level than the agents. In the cooperative setting all foods have a level that is higher than that of a single agent, so the agents have to work together for every single food. The grid size (G), the amount of food available(f), and the amount of agents(N) are all variables. If the agents collect all three rewards, they will get a score of 1. Otherwise they get the percentage of levels of the rewards that they did collect.

RWARE is a cooperative multi-agent environment in which simulates robots moving around boxes in a warehouse. In the environment, agents have to pick up highlighted boxes and deliver them to the goal area. When this is done they have to place the box back into an empty box location. This is a very sparse reward environment, because the agents only get a point after a successful delivery, and after that they still have to replace their box to be able to score new points. The environment is also partially observable, because the agents only have a 3x3 vision range. There are 3 different warehouse sizes to choose from, tiny, small and medium. The amount of agents is set to N. The amount of highlighted boxes(R) is 2N, N or N/2.