# NEURAL MMO: COLLABORATIVE RESOURCE SHARING FOR MULTIPLE FORAGING AGENTS

HARSH GOEL [HARSHG99@SEAS], GAURAV KUPPA [GAKUPPA@SEAS], ADITYA PRATAP SINGH [ADIPRS@SEAS],

ABSTRACT. Multi-agent robotics planning problems for tasks such as environmental monitoring, search and rescue, and surveillance are generally NP hard. In this paper we focus on the multi agent foraging task, where a group of agents collect resources and deposit them at a central depo in a collaborative manner. Existing decentralised solutions to foraging problem are often rule based and don't take into account multiple objectives of exploration and exploitation for foraging, or resource constraints in agents. Our paper's contributions are to apply a reinforcement learning solution to the multi agent foraging problem, which gives us the advantage of making a NP-hard foraging problem computationally feasible,decentralised and scalable to any number of agents under resource constraints.

## 1. INTRODUCTION

Multi-agent robotics planning problems include solving large-scale tasks such as environmental monitoring, search and rescue, and surveillance. In this paper, we focus on the multi agent foraging problem where a team of robots aim to collect resources from the environment and deposit these resources to a central collection spot (Depo or Nest)[1]. For such problems, multiple agents can interact in either cooperative settings, competitive settings or mixed settings[1]. These planning problems are generally NP-hard and many existing solutions are centralized and grow exponentially in complexity with the number of agents. Decentralized solutions to these problems cam often be quite sub-optimal as they rely on rule based approaches from domain based knowledge[2][3]. An effective MAF approach balances the exploration for new resources with the exploitation of already-discovered ones.

In this paper, agents would be deployed in an environment with food and water as shown in Figure 1 and each agent will collect and deposit resources while consuming a few resources to survive. This paper aims to investigate the emergence of behaviour that balances between exploitation of resources for individual survival and the team objective of depositing resources at a Depo through Reinforcement Learning. Learning for Multi agent systems can be difficult to stabilise due to non-stationarity[4] in the environment and as such we apply newer techniques in Reinforcement Learning, specifically Proximal Policy Optimisation (PPO) to train a policy. We design and compare learned policies over two reward structures for the problem.

## 2. BACKGROUND

2.1. **Reinforcement Learning (RL).** A Markov Decision Process (MDP) is characterised by a state of the environment $s_t \in \mathbf{S}$, an action $a_t \in \mathbf{A}$ from a policy $\pi(a_t|s_t)$, transition dynamics $s_{t+1} \approx p(.|s_t, a_t)$ and a corresponding reward $r_t$ for the action $a_t$. RL maximises the long term return $R_t^\pi$ of and MDP at any time step t under policy $\pi$ given by $\sum_{\hat{t}=t}^{\infty} \gamma^{\hat{t}-t} r_{\hat{t}}$. Some algorithms for RL learn a joint mapping $Q(SXA) \rightarrow \mathbf{R}$ that represents the return $R_t$ over all possible configurations of states and actions(Q-Learning)[5][6], or learning a policy $\pi(a_t|s_t)$ that maximises the long term reward (REINFORCE)[7] or both (eg. A2C,A3C)[8]. In the latter a value function and policy over a state are learnt. The value function is used to reduce the variance over the estimated return and by extension over the policy losses.

2.2. **Multi-agent Reinforcement Learning.** Reinforcement Learning for multi-agent systems generally optimise a global objective over a a cooperative game of many agents. In this case each agent $i$ chooses an action $a_t^i \in \mathbf{A_i}$ and the action space is a joint space of all actions for all agents $\mathbf{A} = \times_{i=1}^{n} \mathbf{A_i}$. This agent would also share a portion of the global reward $r(s, u)$. However, the large action space and credit assignment makes centralized RL for multi agent problems infeasible. Thus, typical approaches to multi-agent reinforcement learning (MARL) include making local decisions using a locally defined reward structure. This follows the decentralised training and decentralised execution paradigm where multiple agents are spawned in an environment to sample and learn from individual trajectories [1][9][10]. Asynchronous methods [8] using actor critic methods are typically common place for training a shared policy network among agents, however, training can be difficult to stabilise. Recent advancements such as Trust region methods (TRPO and PPO) [11][12] can be used to stabilise training for multi-agent reinforcement learning problems.

## 3. Related Work

MAF is a widely studied problem for swarm intelligence [1] and involves numerous subtasks such as agent exploration, path planning and the constant need to balance exploitation and co-operation amongst agents. Rule based approaches exploit domain knowledge for a small subset of these problems to arrive at sub-optimal decentralised solutions[2][3]. A large body of work applies reinforcement learning to the the multi agent foraging problem [1][9][13][14][15]. These works typically focus on learning cooperative behaviour to efficiently explore the environment[1][9], balance the exploration vs exploitation aspect of the environment[14][15], or learn communication channels to coordinate agents to search the environment for resources[13]. However, these approaches do not take into account the constraints on agents to perform foraging tasks. Cooperative behaviours can often be influenced when agents need to prioritise their own health and safety, and this paper aims to learn cooperative behaviour under such individual agent constraints. While individual agent constraints could be too harsh to learn useful cooperative behaviour, we introduce automatic resource sharing amongst agents once within a communication range to relax the problem.

## 4. Problem Formulation

4.1. **Environment.** We set up a two-dimensional, grid-world environment of size 32x32, with randomly placed resources and obstacles and a single, central nest/Depo. This is based off from OpenAI's Neural MMO environment[16] as showm in Figure 1. The 2D environment consists of a table of tiles, where each tile can either be a Forest Tile, Scrub Tile, Water Tile, Stone Tile and Depo Tile. The Forest tile contains food that are depleted when an agent walks. A scrub tile provides no resources and is only traversable. Furthermore, depleting a forest tile restores 100% of the food to the agent that walks over this tile. A Scrub Tile has a 2.5% chance to turn into Forest Tile at each time step. Walking adjacent to a Water tile restores 100% water to the player and the tile cannot be depleted and is not traversable. A grass tile is a traversable tile which causes no change in resources. A stone Tile is non traversable. Finally, agents can interact with a Depo Tile to deposit resources into this tile. Once an agent is on Depo Tile, it would automatically deposit half of it's personal resources to the Depo.

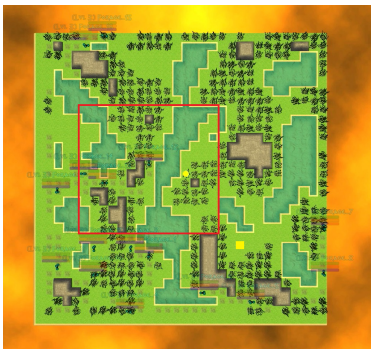


Figure 1. Visualization of Foraging Environment in Neural MMO. The red box depicts the observable region for the agent at the center.

4.2. **Agent.** Each agent (depicted by 3 blue dots in Figure 1) has a maximum capacity for holding resources. Each agent starts with 10 food and 10 water. Agents can collect more resources up to this max capacity as the agent traverses the map and forages more resources. Each agent consumes 1 food and 1 water at each time step. If agents have 0 resources, the health of each agent deteriorates over time. Each timestep without food or water takes away 10 health from a maximum of 100. Health can be restored if the agent has resources above the half the maximum capacity. Furthermore, agents can deposit its foraged resources to the Depo tile. Agents can chose to enter a Depio tile to deposit half of their food and water, agents would loose this food and water for their own personal survival. Additionally, to relax the problem we allow agents to share half their resources amongst each other once they are within two tile distances from another agent.

4.3. **Final Setup.** An environment is setup where a team of 16 agents is spawned around a randomnly allocated Depo Tile. Terrain generation is done using a default octave generator provided by NeuralMMO [16]. An episode is set to terminate if all agents in the team die.

## 5. APPROACH

We attempt to break the Foraging problem to a Markov Decision Process with observations, actions and rewards as described below. We implement an actor critic architecture for training the policy network and use PPO to stabilise the parallelised training over multiple CPU's.

5.1. **Observation Space.** Agents observation are set up to be partially observable with a 15 by 15 field of view ash shown in 1. Each agent has access to it's own state - food and water level, health, position and ID (depth of 12). Besides this, each agent has the same information for the first 100 agents within this field of view. Since we have 16 agents in the problem, only a maximum of 15 rows of the entity observation model will be filled at any time. Observations also include a snapshot of the locally observed environment within the the 15x15 field of view centered around an agent(depth of 4). Each agent has access to the type of tiles (food, water or grass) in said grid and the absolute position of the tile in the environment. Furthermore each agent also has access to the absolute location and total resources in the Depo tile regardless of whether the tile is within the field of view of the agent.

5.2. **Action Space.** Agents can perform 4 actions in the environment to interact with other agents for sharing resource, forage for food and water and deposit resources. These actions result in a movement in each of the four cardinal directions in the two dimensional grid world; each results in a one-cell movement that takes one time step to execute.

5.3. **Reward Structure.** We propose two reward structures that consists of both team and individual rewards for foraging efficiently in the environment. For both settings agents receive a large death penalty if they die and 0 rewards if they manage to survive. All agents in the team are rewarded if any single agent manages to deposit resources in the environment. The agent that deposits the resources gets an additional reward. The team reward and individual reward is linearly proportional to the number of resources as follows:

$$r_t = \lambda_1 F_d + \lambda_2 W_d + \mathbf{I}^{\mathbf{i}}_{\mathbf{Depo}}(\mu_1 F_d + \mu_2 W_d)$$

where $F_d$ $W_d$ are deposited food and water and $\mathbf{I}^{\mathbf{i}}_{\mathbf{Depo}}$ indicates whether the agent is currently on the depo tile(the current agent is depositing food). The table below summarises the coefficients of the reward structures in 2 different settings. One setting learns a policy using only team rewards, the other learns a policy using both team rewards and agent rewards.Note that the depositing food and water are equally weighed.

| Reward | Structure 1 | Structure 2 |
|---|---|---|
| Death | -20 | -20 |
| $\lambda_1$ | 0.05 | 0.05 |
| $\lambda_1$ | 0.05 | 0.05 |
| $\mu_1$ | 0.0 | 0.1 |
| $\mu_2$ | 0.0 | 0.1 |

TABLE 1. Reward Structure

5.4. **Learning.** We use actor critic formulation of the policy gradients to train the network. In this case we have two networks with parameters $\theta$ and $\phi$ that represent the policy (actor) and value function (critic) for the agent. Both the networks have shared parameters except for the last layer. Observations are processed as shown in 2 and are fed to a LSTM for memory. The value and policy is obtained from the LSTM state.

We train the network by sampling episodes on policy. We train the policy and value using asynchronous methods A3C in a distributed hierarchical fashion[8]. In this algorithm, there is a global thread that stores a global copy of the weights of the policy and value networks for each agent. Furthermore, agents share a common learning environment in a worker thread and each agent has its own local copy of a common (global) policy neural network for policy and value estimation. Each epoch samples from this environment an episode with a fixed $N$ sequence of transitions $s_t$ $a_t$, $r_t$, $s_{t+1}$ for each agent using this local copy of the policy parameters. Since some agents can die due to the environment definition, we store dummy observations and rewards for those transitions and these are neglected during gradient computation.

The experience buffer is then passed to the global network for gradient computation and network update using the PPO algorithm with the following losses. The value network is trained with the MSE loss as follows:

$$L^V(\phi) = \hat{\mathbb{E}}\left[R(s_t) - V(s_t;\phi))^2\right]$$

where $R(s_t)$ is the total discounted return from state t in the sampled trajectory.

$$R(s_t) = \hat{\mathbb{E}} \sum_{i=t}^{N} \gamma^{i-t} r_i$$

Since the training is asynchronous, at any the time global copy of the policy weights could be different from the weights that were used to sample an experience buffer. Thus, the policy network is updated by using a clipped objective function with PPO to prevent updates to the global policy network from samples that are far away from the policy. This helps stabilise training over multiple CPUs and GPUs. The objective function is as follows:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}_t} \left[ min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon) \right]$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

where $\epsilon$ = 0.3. $\hat{A}_t$ is the generalised advantage function to reduce the variance in the policy gradients. It is defined as follows:

$$\hat{A}_t = \sum_{i=t}^{N} (\lambda)^{i-t} \delta_t$$

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

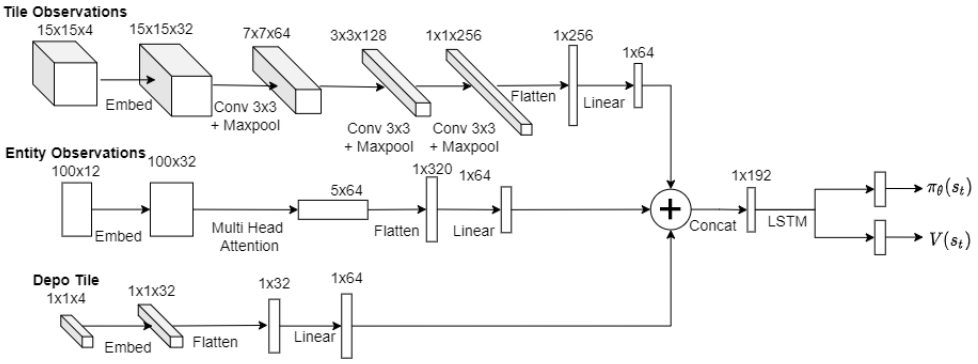


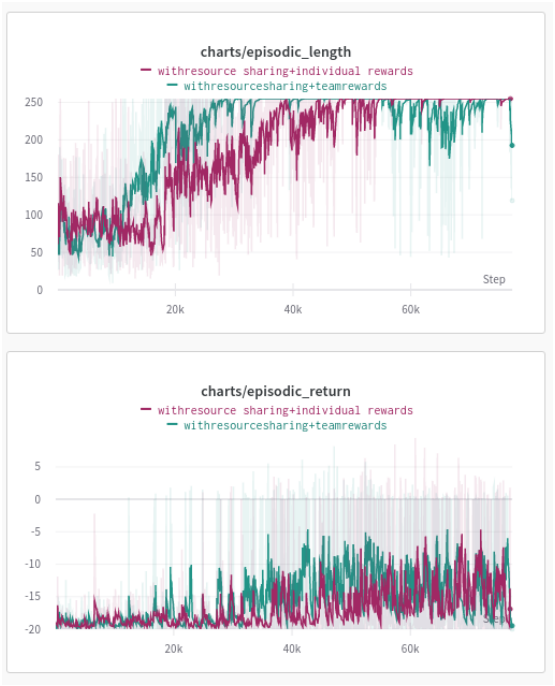

FIGURE 2. Visualization of Foraging Environment in Neural MMO

## 6. EXPERIMENTAL RESULTS

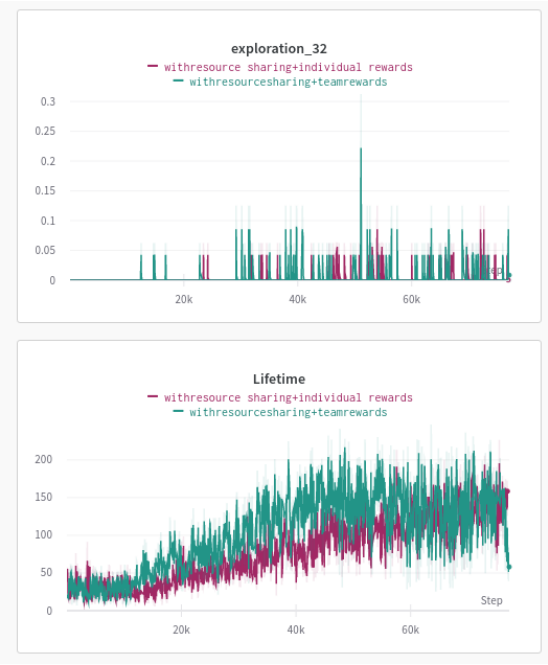
6.1. **Training Details.** We train 2 models for the above-mentioned reward structures on a server where each model is trained over 64 CPU cores and 1 V-100 GPU. The models are trained for 75000 episodes over 2 days where each episode collects an experience buffer of maximum 256 timesteps. The data is collected parallely over the 64 cpu cores over a version of the network parameters and the global network parameters are asynchronously updated with the collected experience buffer once a worker thread completes an episode.

6.2. **Metrics.** We study the performance of the two reward structures with the implemented training architecture. We report the following metrics:

(1) Average return per agent: Cumulative reward per agent per episode. It indicates whether the policy is training and agents aren't learning random behaviour.
(2) Episode length for the team: Average timesteps per episode till all agents in the team die. Indicates resource collection potential of the team (more episode time available to maximise resources deposited).
(3) Average agent lifetime: Average life expectancy using the current policy. It indicates the whether the policy learns behaviour that exploits the environment for individual benefit.
(4) Environment Exploration (after 32*4 timesteps): ratio of number of tiles seen in the environment to total traversable tiles. Indicates team performance of exploring to find more resources to collect and deposit.




(A) Episode Length and Return




(B) Exploration Rate and Agent Lifetime

FIGURE 3. Training Results(Team rewards -Structure 1 only in Green, Team with Individual Rewards - Structure 2 in Purple

6.3. **Analysis.** From the episodic returns curve, it is clear that both models learn some some behaviour for depositing resources that is better than a random policy. The policy learnt through reward structure 1 shows to have better average exploration compared to reward structure 2. This can be explained by the fact that by assigning individual rewards, agents are incentivised to collect and deposit nearby resources to the depo from the environment and not move closer to other agents to share resources. Due to the the finite lifespan, maximum resource capacity and resource consumption, this behaviour wouldn't allow for exploring the environment. This has a subtler consequence, as food resources are exhaustible, agents would quickly exhaust nearby food resources and would die off quickly due to their internal constraints. This leads to a slightly worse average lifetime per agent for the policy with reward structure 2 as shown in Figure 3b.

After 75,000 interactions, policy from reward structure 2 seems to have higher team episode length, indicating that one agent is more likely to survive in the team with individual rewards. This has a subtle explanation and we think that this is due to the higher variance in terms of returns and average lifetime. This is introduced due to conception of team rewards, since individual agents are rewarded if any agent on the team deposits resources, agents that perform random behaviour might get rewarded even though their actions didn't contribute to the team through resource sharing. While this may help with exploration, it is likely to destabilise training as policy gradients would try improving both explorative behaviour and exploitative behaviour without context, and lead to a policy that is sub-optimal.

## 7. CONCLUSIONS

In this paper, we leverage recent advances in distributed learning to propose a reinforcement learning solution to multi-agent foraging with agent constraints. Our learned approach, endows agents with the ability to balance exploitation and exploration to an extent using only team based rewards, but further improvements are necessary to stabilise the training for situations where the network optimises explorative and exploitative behaviour without context.

This paper does show that policies can be learnt for np-hard multi agent problems in a decentralised manner that may have multiple objectives and subtasks. Solutions that involve training decentralised policies are scalable and this approach can be deployed on individual robots hence making it robust as compared to a centralised controller which for many current multi robotic systems in industry is a critical point of failure.

The code can be found at https://github.com/gauravkuppa/neural_mmo/tree/cleanrl.

## REFERENCES


[1] M. Yogeswaran, S. Ponnambalam, and G. Kanagaraj, "Reinforcement learning in swarm-robotics for multi-agent foraging-task domain," in Symposium on Swarm Intelligence (SIS). IEEE, 2013, pp. 15–21

[2] O. Zedadra, H. Seridi, N. Jouandeau, and G. Fortino, "A cooperative switching algorithm for multi-agent foraging," Engineering Applications of Artificial Intelligence, vol. 50, pp. 302–319, 2016.

[3] L. Panait and S. Luke, "A pheromone-based utility model for collaborative foraging," in Proceedings of AAMAS. IEEE, 2004, pp. 36–43

[4] Papoudakis, G., Christianos, F., Rahman, A., Albrecht, S. V. (2019). Dealing with non-stationarity in multi-agent deep reinforcement learning. arXiv preprint arXiv:1906.04737.

[5] Sutton, Richard; Barto, Andrew (1998). Reinforcement Learning: An Introduction. MIT Press

[6] Van Hasselt, Hado; Guez, Arthur; Silver, David (2015). "Deep reinforcement learning with double Q-learning" (PDF). AAAI Conference on Artificial Intelligence: 2094–2100. arXiv:1509.06461

[7] Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." Advances in neural information processing systems 12 (1999).

[8] Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." International conference on machine learning. PMLR, 2016.

[9] Christianos, F., Papoudakis, G., Rahman, M. A., Albrecht, S. V. (2021, July). Scaling multi-agent reinforcement learning with selective parameter sharing. In International Conference on Machine Learning (pp. 1989-1998). PMLR.

[10] Jin, C., Liu, Q., Wang, Y., Yu, T. (2021). V-Learning–A Simple, Efficient, Decentralized Algorithm for Multiagent RL. arXiv preprint arXiv:2110.14555.

[11] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. "Trust region policy optimization". In: CoRR, abs/1502.05477 (2015).

[12] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[13] Shaw, S., Wenzel, E., Walker, A., Sartoretti, G. (2022). ForMIC: Foraging via Multiagent RL With Implicit Communication. IEEE Robotics and Automation Letters, 7(2), 4877-4884.

[14] Hahn, C., Ritz, F., Wikidal, P., Phan, T., Gabor, T., Linnhoff-Popien, C. (2020, July). Foraging swarms using multi-agent reinforcement learning. In ALIFE 2020: The 2020 Conference on Artificial Life (pp. 333-340). MIT Press.

[15] Yogeswaran, M., Ponnambalam, S. G., Kanagaraj, G. (2013, April). Reinforcement learning in swarm-robotics for multi-agent foraging-task domain. In 2013 IEEE Symposium on Swarm Intelligence (SIS) (pp. 15-21). IEEE.

[16] Suarez, J., Du, Y., Isola, P., Mordatch, I. (2019). Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents. arXiv preprint arXiv:1903.00784.