Exploring Reward Sharing Strategies for Effective Cooperative Multi-Agent Task Completion

Juji Lau, Sanjana Nandi
(jal499, sn523)
(Group 7)
https://github.com/juji-lau/on-policy/tree/main

1 Introduction

In many real-world situations where the task is too big for a single actor to accomplish alone, effective cooperation among multiple agents is crucial for success. A few examples of such situations include: search and rescue missions, hazardous environment cleanup, and area surveillance.

Currently, policy gradient methods such as Proximal Policy Optimization (PPO) have proven successful for single-agent settings. PPO is a widely used policy gradient algorithm that focuses on updating the policy of individual agents based on local observations, and individual rewards. In PPO agents act independently, which works well for single-agent tasks (such as learning to walk, moving stationary objects, navigating a messy environment, etc.). However, in tasks requiring coordination among multiple agents, the lack of global context, and therefore inter-agent communication, may lead to suboptimal behaviors.

This is where Multi-Agent Proximal Policy Optimization (MAPPO) steps in. The MAPPO algorithm modifies the PPO algorithm to better suit multi-agent environments. In MAPPO, a single critic aggregates the observations and actions from all acting agents to compute value estimates while each agent maintains its own policy network. This shared critic allows indirect "communication" between agents, providing them a common understanding of the environment and of each others' actions. As a result, MAPPO agents can make decisions that better account for team-level objectives, leading to more coherent group strategies, and hence an overall better task outcome.

2 Problem

While the structure of MAPPO inherently encourages multi-agent cooperation, its true success in a cooperative setting depends heavily on how rewards are structured. This leads us to our research question:

How do we best structure the reward function to maximize the collective performance of agents in a cooperative environment under the MAPPO framework?

There are numerous possible reward functions. For this project, we grouped the possibilities into three main categories. (Let r_i denote the reward given to an individual agent.):

1. Purely Individual Rewards: $r_i = R_i$
2. Partially Shared Rewards: $r_i = R_{ps} =$ Some combination of R_i and R_s
3. Entirely Shared Rewards: $r_i = R_s$

We aim to determine which category of reward function (or combination of categories) leads to the most effective task completion in multi-agent, cooperation-dependent settings. We hypothesize that in such a setting, agents trained with Partially Shared Rewards (R_{ps}), will achieve better task coverage and cooperation than agents trained with Purely Individual Rewards (R_i), which in turn will outperform agents trained with Entirely Shared Rewards (R_s).

3 Approach

To test our hypothesis, we trained agents with the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm using reward function(s) from each of the three categories. This is link to our implementation: https://github.com/juji-lau/on-policy/tree/main

3.1 Experimental Setup: Environment

For our multi-agent cooperative environment, we used the Simple Spread environment from the PettingZoo library's Multi-Agent Particle Environments (MPE) [1][2]. Simple Spread is a cooperative game where N agents must attempt to cover N stationary landmarks in t time-steps without colliding. We chose Simple Spread because it's a *fully* cooperative task (i.e., there are no competitive elements), which encourages all agents to work towards a common goal. This setup allows us to compare the effectiveness of different reward structures without confounding factors.

3.2 Experimental Setup: Training

To maximize the models' performance under their given reward function, while ensuring a fair evaluation, all models were trained with the following parameters:

```
--num_env_steps 100000
--episode_length 25
--ppo_epoch 30                      # previously 15
--clip_param 0.1                    # previously 0.2
--entropy_coef 0.005                # previously 0.01
--lr 5e-4                           # unchanged
--critic_lr 5e-4                    # unchanged
--use_valuenorm                     # added to source code
--use_feature_normalization         # added to source code
--hidden_size 128                   # added to source code
--layer_N 2                         # added to source code
```

If not listed above, we used the defaults provided by the MAPPO implementation [1][2]. These parameters gave the best average reward and entropy across all three reward schemes.

Additionally, the number of landmarks and agents is set at ($N = 3$) for all models.

3.3 Experimental Setup: Reward Structures

To test our hypothesis about how reward structure influences cooperative task performance under MAPPO, we modified the official implementation of MAPPO [3] to train three variants, each using one of our defined reward schemes. The reward function for each structure is given below:

1. *Purely Individual Rewards*: The rewards for each agent are based purely on individual performance. (This encourages selfish behavior).

```
def individual():
    ideal_dist = 0.25
    tolerance = 0.1
    bonus = 1.0
    penalty_close = -1.0
    penalty_far = -0.5
    collision_penalty = -1.0

    # Find nearest landmark
    distances = [np.linalg.norm(agent.state.p_pos -
                l.state.p_pos) for l in world.landmarks]
    min_dist = min(distances)

    # Reward shaping
    if abs(min_dist - ideal_dist) < tolerance:
        proximity_reward = bonus
    elif min_dist < ideal_dist:
        proximity_reward = penalty_close
    else:
        proximity_reward = penalty_far

    # Penalize collisions
    collision_loss = 0
    if agent.collide:
        for other in world.agents:
            if other is not agent
                and self.is_collision(agent, other):
                collision_loss += collision_penalty

    return proximity_reward + collision_loss
```

2. *Partially Shared Rewards*: The rewards for each agent are dependent on both individual performance and collective progress. (This encourages a mix of selfish behavior and global coordination).

```
def partially_shared():
    # Individual component
    individual_rewards = []
    for agent in world.agents:
        dists = [np.linalg.norm(agent.state.p_pos -
                l.state.p_pos) for l in world.landmarks]
        min_dist = min(dists)
        individual_rewards.append(-min_dist)

    # Shared component (mean distance to landmarks)
    total_dist = 0
    for l in world.landmarks:
        for a in world.agents:
            total_dist += np.linalg.norm(a.state.p_pos -
            l.state.p_pos)

    if not hasattr(world, "prev_total_dist") or
    world.prev_total_dist is None:
        world.prev_total_dist = total_dist

    shared_progress = world.prev_total_dist - total_dist
    world.prev_total_dist = total_dist

    # Coverage bonus: reward if each landmark is closest to a
    different agent
    closest_agents = [np.argmin([np.linalg.norm(a.state.p_pos -
        l.state.p_pos) for a in world.agents]) for l in
        world.landmarks]
    coverage_bonus = len(set(closest_agents))  # Higher if
```

```
        agents spread out

        return 0.6 * individual_rewards[world.agents.index(agent)]
        + 0.4 * (shared_progress + coverage_bonus * 0.1)
```

3. *Entirely Shared Rewards*: All agents receive the same reward, which is a function of collective progress. (This encourages global coordination).

```
    def shared():
        total_dist = 0
        for l in world.landmarks:
            for a in world.agents:
                total_dist += np.linalg.norm(a.state.p_pos - l.state.p_pos)

        if not hasattr(world, "prev_total_dist") or world.prev_total_dist is None:
            world.prev_total_dist = total_dist

        progress = world.prev_total_dist - total_dist
        world.prev_total_dist = total_dist

        # Penalize collisions
        collision_penalty = 0
        for i, a1 in enumerate(world.agents):
            for j, a2 in enumerate(world.agents):
                if i != j and self.is_collision(a1, a2):
                    collision_penalty -= 1

        return progress + collision_penalty
```

3.4 Experimental Setup: Evaluation

To provide a consistent baseline comparison and to ensure a fair evaluation, the performance of all models were evaluated using Simple Spread's original reward function:

```
def reward(self, agent, world):
    # Agents are rewarded based on minimum agent distance to each landmark:
    rew = 0
    for l in world.landmarks:
        dists = [np.sqrt(np.sum(np.square(a.state.p_pos - l.state.p_pos)))
                 for a in world.agents]
        rew -= min(dists)

    # Agents are penalized for collisions:
    if agent.collide:
        for a in world.agents:
            if self.is_collision(a, agent):
                rew -= 1
    return rew
```

In this function, agents are globally rewarded by summing the distance between each landmark and its closest agent (encouraging coverage) and locally penalized for each collision with another agent (encouraging inter-agent cooperation).

Technically agents are penalized based on their distances from the landmarks, but the statement above is conceptually correct

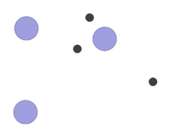
Figure 1: **Purely Individual Rewards** This figure depicts the final landmark coverage of *N*=3 agents trained under the *purely individual reward* function.

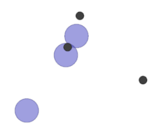
Figure 2: **Partially Shared Rewards** This figure depicts the final landmark coverage of *N*=3 agents trained under the *partially shared reward* function.

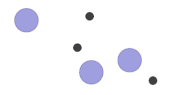
Figure 3: **Entirely Shared Rewards** This figure depicts the final landmark coverage of *N*=3 agents trained under the *entirely shared reward* function.

4 Results

The qualitative and quantitative results across all three rewards schemes reveal that task completion and group coordination improves as the reward structure becomes more shared.

4.1 Qualitative Results

The final spatial coverage achieved by 3 agents after 25 time steps in the Simple Spread environment are depicted in Figures 1, 2, and 3.

Although none of the reward schemes resulted in all of the landmarks being covered, the partially shared reward scheme came the closest, followed by entirely shared reward scheme, and then the individual reward scheme.

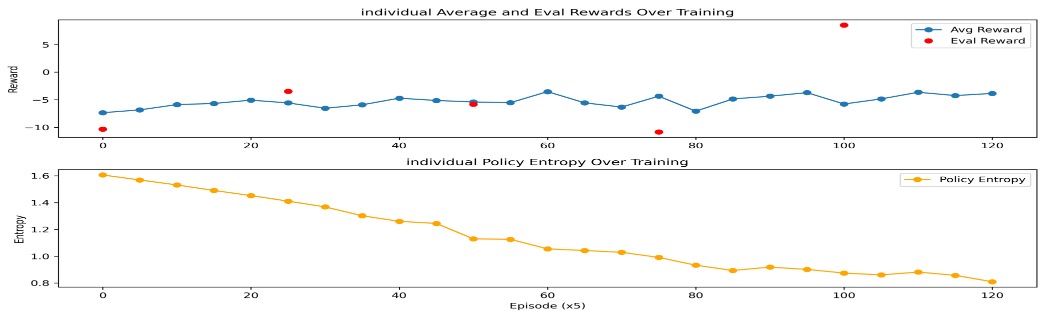

Figure 4: **Purely Individual Rewards** This figure depicts the average training and evaluation rewards, along with the entropy of *N*=3 agents trained under the *purely individual reward* function.

With the individual reward scheme, the agents optimized their own rewards and often converged on the same landmarks, resulting in poor coverage and coordination. The partially shared reward scheme lead to an attempt at unsuccessful coordination as a result of conflicting reward signals from the individual and shared components. The shared reward scheme produced the most effective group behavior: the agents tried to distribute themselves across landmarks with minimal collisions. This seems to suggest that shared rewards can encourage cooperative behavior and more efficient task completion in multi-agent situations.

4.2 Quantitative Results and Analysis

The average training rewards, average evaluation rewards, and entropies, for the individual, partially, and fully shared rewards are plotted in Figures 4, 5, and 6 respectively.

The maximum entropy for an choosing among *x* discrete actions, is given by:

$$H_{max} = log(x)$$

In Simple Spread, agents had *5* actions to choose from:

1. No action
2. Go up
3. Go down
4. Go left
5. Go right

Meaning the maximum entropy for our agents is given by $H_{max} = log(5) = 1.609$. Hence, we can use the following benchmarks to interpret the model entropy:

1. $h_{model} = 1.6$: the policy is still random and exploratory
2. $h_{model} = 1.0$: the policy is starting to prefer certain actions
3. $h_{model} = 0.5$: the policy is very confident, often choosing the same actions

As shown in the graphs, all reward types experience a downward trajectory of entropy as the model trains. It appears that the *less* individualistic the reward, the *faster* the entropy converges to zero. In other words, individualism *encourages* model exploration.

In the Simple Spread environment, higher reward values indicate more successful task performance. Based on our graphs, the *Entirely Shared Rewards* performed the best, followed by the *Partially Shared Rewards*, which performed about the same as the *Purely Individual Rewards*. This was surprising as we predicted that the *Partially Shared Rewards* would outperform the *Purely Individual Rewards*, which would outperform the *Entirely Shared Rewards*.

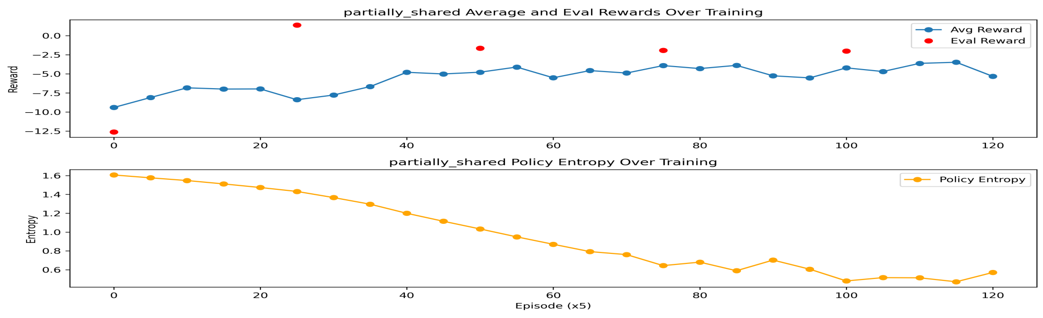

Figure 5: **Partially Shared Rewards** This figure depicts the average training and evaluation rewards, along with the entropy of N=3 agents trained under the *partially shared reward* function.

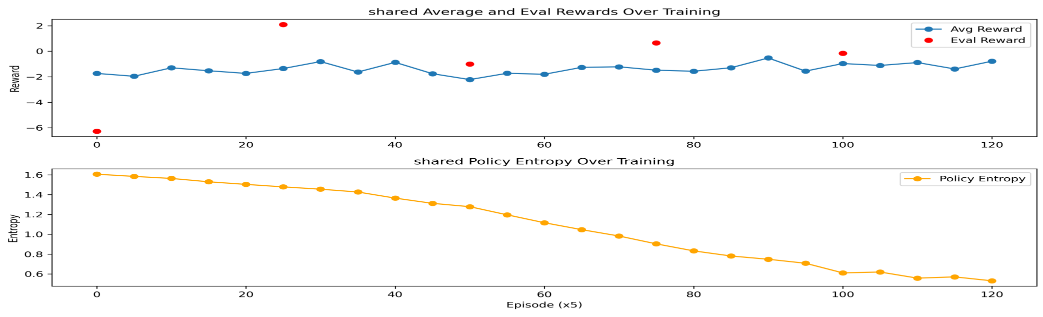

Figure 6: **Entirely Shared Rewards** This figure depicts the average training and evaluation rewards, along with the entropy of N=3 agents trained under the *entirely shared reward* function.

When we compare average returns and policy entropy convergence across all three reward schemes, a pattern emerges: the greater the emphasis on shared rewards during training, the better the task completion in a multi-agent cooperative setting. This is evidenced by the fact that the model trained with *Entirely Shared Rewards* outperformed the model trained with *Partially Shared Rewards*, followed *extremely* closely by the model trained with *Purely Individual Rewards*.

4.3 Refining Process

Originally, our rewards were identical for all three reward structures, and stuck at around -136. After a lot of debugging, we changed a specific configuration in the code, which differentiated the rewards and increased them all to around -50.

We also noticed that our entropy stayed at around 1.5, indicating that there wasn't much policy exploitation. To remedy this, we increased the *ppo epoch* parameter from 10 to 20, and decreased the *entropy coef* parameter from 0.01 to 0.005. This gave the agents more time to explore (and therefore decide on a suitable policy), while simultaneously placing *less* weight on exploration (thus implicitly more weight on exploitation). This improved our rewards from -50 to around -35.

4.4 Future Steps

Given more time, we would have liked to further train our model for all three reward structures until their average reward is closer to zero, and their entropy is less than 0.5.

5 Individual Contribution

5.1 Juji Lau (jal499)

Juji took primary responsibility in organizing, drafting, and polishing the project proposal and final report. She also contributed to the implementation including coding, debugging and refactoring. Together, her and Sanjana contributed equally to developing research ideas, defining reward functions, and determining which metrics to collect. Both contributed equally to planning and producing the presentation video, as well as evaluating other groups' submissions.

5.2 Sanjana Nandi (sn523)

Sanjana took primary responsibility in modifying the official implementation of MAPPO [3], training the model, and gathering data. She also contributed to drafting and polishing the project proposal and final report. Together, her and Juji contributed equally to developing research ideas, defining reward functions, and determining which metrics to collect. Both contributed equally to planning and producing the presentation video, as well as evaluating other groups' submissions.

References

[1] I. Mordatch and P. Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

[2] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

[3] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.