Appendix for the Melting Pot Contest 2023 Report

A Further Results and Analysis

In this section, we provide extended analysis of the contest results including agent behaviors and provide more details on the capability analysis.

A.1 Performance analysis for each substrate

Figure 7 provides a detailed view of performance of each of top 12 teams on each substrate. It further compares the difference between performance of focal agents vs the return achieved by the background agents for each substrate. This comparison lends further support to the point of focusing on different metrics for evaluating cooperative intelligence in addition to the mean focal return used in this contest.

A.2 Assessing robustness of contest score

While our analysis has helped establish the merits considering various evaluation metrics on the top of focal per capita return, it is known that 'mean' is an aggregation technique often dominated by outlier performance. Following [34], in this section, we assess the validity of team rankings when measured with more robust metrics. Specifically, we consider **Inter-Quartile Mean:** IQM discards the bottom and top 25% of the runs and calculates the mean score of the remaining 50% runs. In our case, we consider IQM over episodes. Figure 8 shows that the team rankings under mean focal return holds when measured using IQM (although they are much closer under IQM), thereby lending confidence in robustness of overall rankings of the contest.

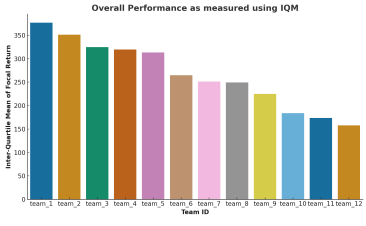


Figure 8: Overall Performance as measured using IQM

A.3 Another example of in-group coordination: CleanUp

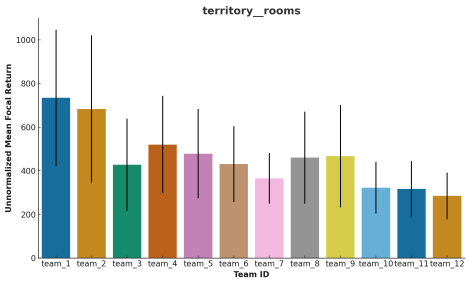
In section 3.3, we performed qualitative analysis of agent behaviors in a case where agents learned to perform in-group coordination leading to anti-social outcomes. In this section, we analyze another such use case, where agents again perform in-group coordination but with an intention to analyze the behavior of other agents and respond accordingly (not leading to an anti-social outcome necessarily). Specifically, in Figure 9, the scenario is such that background agents would reciprocate by cleaning the river if one focal agent starts to clean the river. In this case, the focal agents form a line at different places in the map to recognize in-group agents and then they use this information to initiate cleaning of river by background agent while switching to always consuming apples themselves.

A.4 Details on capability profiles, measurement layouts and their predictive power

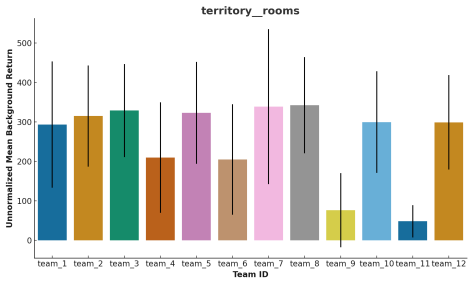
To build the evaluation models, we use the experimental data containing nine features: submission ID, four substrates ($territory\_\_rooms$, $allelopathic\_harvest\_\_open$, $clean\_up$, and $prisoners\_dilemma\_in\_the\_matrix\_\_arena$) and four tags ($resident$, $visitor$, $defense$, and $flexibility$). We also use the output feature, the 'Normalized Score'. To help with the interpretation of results and the output distribution of the measurement layout, we min-max scale these scores in the dataset between 0 and 1.

As described in the main paper, the first model we fit is an assessor, an XGBoost regression model from the XGBoost python library, considering 51 cases per submission (aggregated for all episodes). Fig. 10 shows the feature importance given from the library. We find that $visitor$, $clean\_up$ and $territory\_\_rooms$ as the features with highest importance.

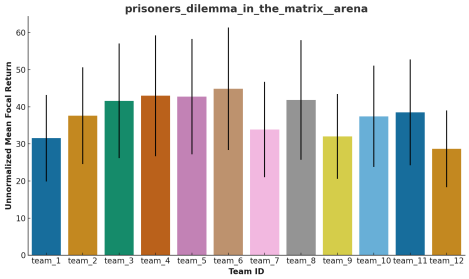
The measurement layouts are built independently for each submission. So we do not use the submission Id as in the assessor model. The measurement layout basically uses these nine features, now considered as task demands, and performance of agents to infer values for each corresponding



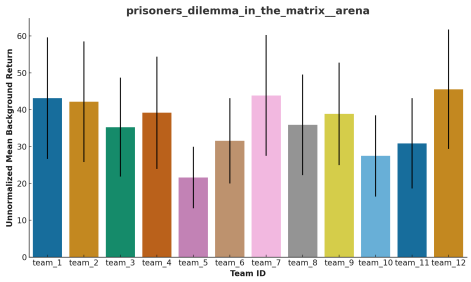
(a) Territory Rooms (Focal)



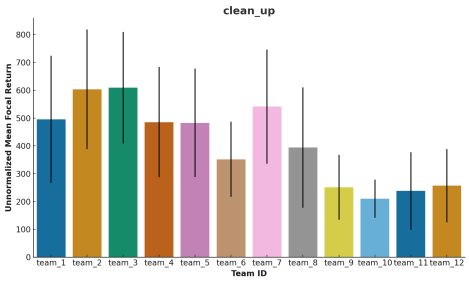
(b) Territory Rooms (Background)



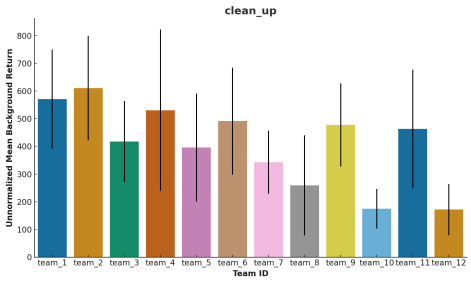
(c) Prisoner's Dilemma in the Matrix Arena (Focal)



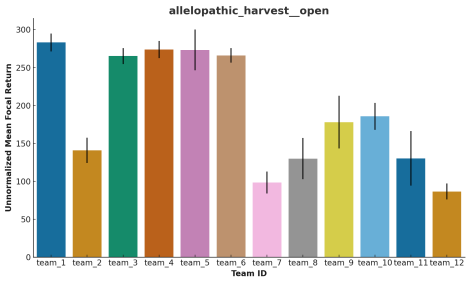
(d) Prisoner's Dilemma in the Matrix Arena (Background)



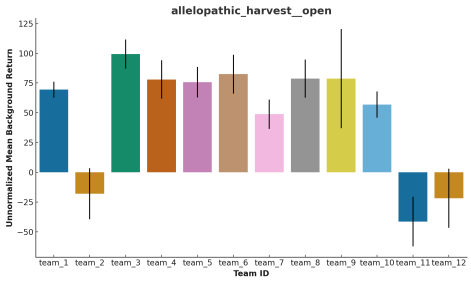
(e) Clean Up (Focal)



(f) Clean Up (Background)



(g) Allelopathic Harvest Open (Focal)



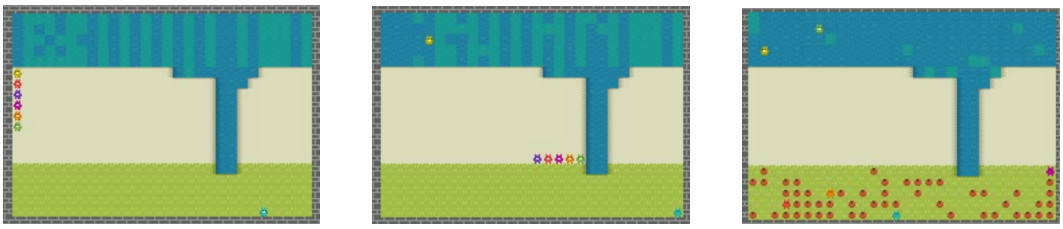
(h) Allelopathic Harvest Open (Background)

Figure 7: Comparison of Mean Focal and Background Return for different substrates

capability. See Figure 11. For more information about how to interpreting these hierarchical Bayesian networks, we refer the reader to [24].

The measurement layout in the figure shows eight abilities: allelopathicHarvestOpenAbility, cleanUpAbility, prisonersDilemmaInTheMatrixArenaAbility, territoryRoomsAbility, residentAbility, visitorAbility, defenseAbility and flexibilityAbility.

Each ability corresponds to a demand. Abilities are compared against demands, their difference is then pass through a logistic function. The outputs of the logistic functions are multiplied together with a noise variable, to model the observed performance.



(a) Clean Up stage 1 (b) Clean Up stage 2 (c) Clean Up stage 3

Figure 9: Progression of Agent behavior in Clean Up game

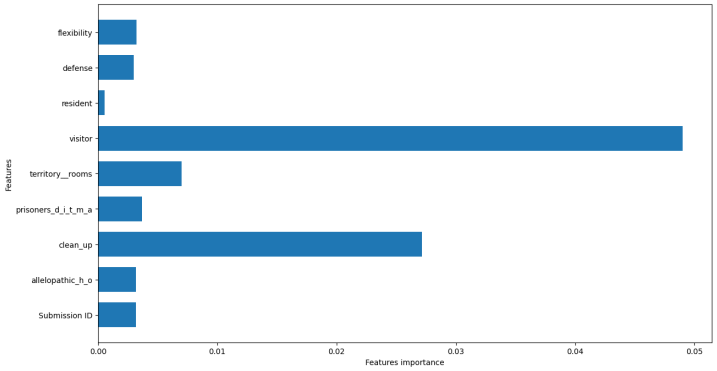


Figure 10: The average feature importance with the assessor and 408 rows.

We divide the dataset into 90% training data and 10% test data and repeat the experiment three times, calculating the means, seen in Fig. 4. We see that prisonersDilemmaInTheMatrixArenaAbility and allelopathicHarvestOpenAbility have a low value for all submissions. Individual team values on cleanUpAbility vary significantly, with $team2\_id5$ showing only 1.28 and $team4\_id16$ showing the highest at 2.22. On both residentsAbility, the values are relatively higher.

Finally, to validate the measurement layouts, we can look at their predictive power in terms of how well they predict performance for each submission and case in the test set. We use mean squared error (MSE) as metric. We compare the results of the assessor, the measurement layout and the aggregate mean (AggMean). AggMean is the aggregate performance for each team, i.e. a simple the success ratio on the training set. This is the common approach used for evaluating AI/ML systems (calculating and extrapolating test performance ood). The measurement layouts predict using the top-down inference of the constructed cognitive model (the graph seen in Fig. 11). The low values of MSE for the assessor support the inferred capabilities, and are much better than AggMean in predictive power (AggMean has perfect calibration but no refinement), but worse than the assessor. The assessor model, which employs XGBoost to predict performance, is not interpretable, and does not give us capability profile.

B Substrates, Scenarios and Evaluation

In this section, we outline the details of the substrates, scenarios and evaluation setup used for the contest.

B.1 Substrate Wrappers

As a part of our starter-kit, we provided baseline implementation of agents in RLLib for particpants to use as a base for their submissions. In order to make the melting pot environment compatible

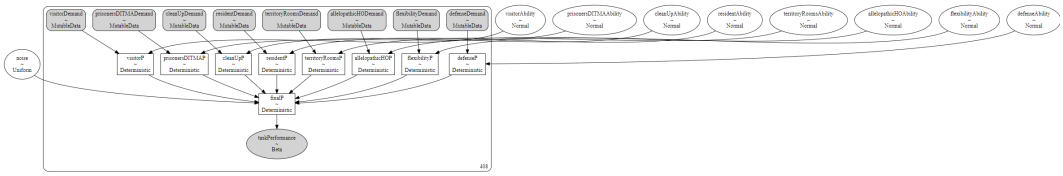
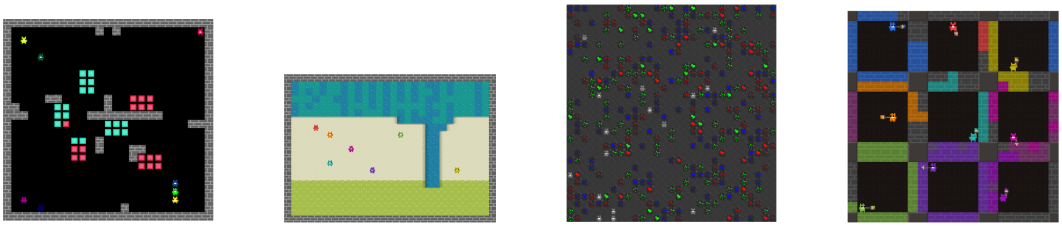


Figure 11: The measurement layout using nine features. The plate says 408 examples, but this is run separately for each of the 8 submissions, so only 51 examples are considered in each plate.

Table 2: The MSE for AggMean, Measurement Layout and XGBoost assessor.

	AggMean	Layout	Assessor
$team\_1\_id\_3$	0.048	0.032	0.014
$team\_2\_id\_6$	0.070	0.040	0.025
$team\_4\_id\_16$	0.071	0.037	0.026
$team\_3\_id\_12$	0.063	0.045	0.014
$team\_5\_id\_8$	0.024	0.014	0.011
$team\_8\_id\_18$	0.027	0.014	0.003
$team\_6\_id\_10$	0.017	0.017	0.007
$team\_11\_id\_26$	0.120	0.012	0.005
average	0.042	0.025	0.012



(a) PD: Arena (b) Clean Up (c) Allelopathic Harv. (d) Territory: Rooms

Figure 12: Substrates considered in the Melting Pot Contest

and extensible with Rllib, we provided an interfacing wrapper[10] that provide a mapping from any MeltingPot substrate to Rllib MultiAgentEnv class and implements the necessary step() and reset() functions. Next, as noted in Section 2.1, all substrates are pixel-based and have sprite of size $8 \times 8 \times 3$ with an observation window of $88 \times 88 \times 3$. Figure 12 provides pictorial overview of the 4 substrates considered in the contest. However, during our experiments, we found that this led to slower training time on the selected environments. To address this, we bench marked the results after reducing the sprite size by a factor of 8 i.e. reducing the observation window to $11 \times 11 \times 3$. We did not find a significant difference in the performance after downsampling the sprite size and hence we provided a wrapper to downsample the sprite as a part of our starter kit. While this downsampling is done at the substrate level, the evaluation on scenario is done on contest servers (described below), where the environment always returns the observations with the full sprite size. To address this, we further provide a downsampling wrapper as a part of agent's policy such that the agent submitted by the participants always receives the downsampled sprite. These adaptations enabled support for effectively and efficiently training agents in low compute regimes, thereby attracting submissions across different academic labs and independent researchers and increasing overall participation.

B.2 Details on Scenarios

Since Melting Pot seeks to isolate social environment generalization, it assumes that agents are familiar with the substrate. They have unlimited access to it during training. The evaluation scheme concentrates on assessing the ability of agents (and populations composed thereof) to cope with the

[10] https://github.com/rstrivedi/Melting-Pot-Contest-2023/blob/main/baselines/wrappers/

presence of unfamiliar individuals in the background population (in a zero-shot way), typically mixed in with familiar individuals from the population under test. A scenario is defined as the combination of substrate and background population of agents. When evaluating on scenario, the focal agents are sampled from the ones submitted by the participants while the background agents are the ones pre-trained using puppet based training pipeline described in the original paper. Scenarios are broadly categorized across two modes: If the mixture contains more focal individuals than background individuals, it is called a *resident* mode scenario. Whereas, if the mixture contains more background individuals than focal individuals, it is called a *visitor* mode scenario. As described in Section 2.4, we provided participants with access to a total of 22 scenarios across the 4 substrates during the development phase. And then we had generated 51 new scenarios that were kept as held-out set for final evaluation. These held0out scenarios were from the same distribution as development scenarios but had variations in some aspects (e.g. visitor vs resident modes, no. of focal vs background agents, reciprocating agents vs convention following agents or a mixture of both etc.). While all the scenarios (including the ones used as held-out set in the contest) will be made available publicly on the base repository[11], we exemplify the scenarios by outlining the ones available during development phase in Table 3.

Table 3: Scenarios used for Development Phase evaluation in Melting Pot Contest 2023

Substrate	Scenario	Description
Allelopathic Harvest	SC 0	Visiting a population where planting green berries is the prevailing convention
	SC 1	Visiting a population where planting red berries is the prevailing convention
	SC 2	Focals are resident and visited by bots who plant either red or green
Clean Up	SC 0	Visiting an altruistic population
	SC 1	Focals are resident and visitors ride free
	SC 2	Visiting a turn-taking population that cleans first
	SC 3	Visiting a turn-taking population that eats first
	SC 4	Focals are visited by one reciprocator
	SC 5	Focals are visited by two suspicious reciproca- tors
	SC 6	Focals are visited by one suspicious recipro- cator
	SC 7	Focals visit resident group of suspicious re- ciprocators
	SC 8	Focals are visited by one nice reciprocator
PD in the matrix: Arena	SC 0	Visiting unconditional cooperators
	SC 1	Focals are resident and visited by an unconditional cooperator
	SC 2	Focals are resident and visitors defect unconditionally
	SC 3	Visiting a population of hair-trigger grim reciprocator bots (a)
	SC 4	Visiting a population of two-strikes grim reciprocator bots (b)
	SC 5	Visiting a mixed population of k-strikes grim reciprocator bots (c)
Territory: Rooms	SC 0	Focals are resident and visited by an aggres- sor
	SC 1	Visiting a population of aggressors
	SC 2	Focals are resident, visited by a bot that does nothing
	SC 3	Focals visit a resident population that does nothing

(a) Bots who initially cooperate but, if defected on once, will retaliate by defecting in all future interactions
(b) Bots who initially cooperate but, if defected on twice, will retaliate by defecting in all future interactions
(c) Bots with k values from 1 to 3, they initially cooperate but, if defected on k times, they retaliate in all future interactions

B.3 AICrowd Evaluation Setup

The participants were required to train the focal agents locally with no restrictions on the training approach. For evaluation, AICrowd services were used where the evaluators were hosted on multiple compute nodes. Specifically, the participants were required to submit a population of n agents (where $n = 1$ was allowed) using the code provided in the baseline repository[12] and AICrowd submission

[11] https://github.com/google-deepmind/meltingpot
[12] https://github.com/rstrivedi/Melting-Pot-Contest-2023

kit[13]. Once the agent(s) were submitted, for each episode, the evaluator would sample focal agents from the submitted population as needed by a given scenario. During the development phase, each participant was allowed to make 2 submissions per day (with an allowance of 5 failed submissions). The participant's submitted code was packaged into a Docker image for every submission, and run on AWS. Each submission was provided with 1 vCPU and 3 GB RAM per focal agent. The resources were typically by soft-constrained using the Ray framework, while running all the agents on a single machine with 8 or 16 vCPUs, depending on the scenario. During development phase, for each scenario, the population needed to complete 4 episodes in 15 minutes. No inter-agent communication is allowed (apart from using the actions to implicitly communicate in the observation space). For practice generalization phase, the evaluation was done for 20 episodes. And for the final evaluation phase, the evaluation was conducted for 80 episodes to account for the variance in the submissions.

Further, AICrowd hosted the leaderboard to show the scores at different granularity levels to the participants. During development phase, the score were computed through evaluation on public scenarios which participants also had accessed to. During practice generalization phase (which ran for 10 days), participants received a score computed on a small sample of held-out scenarios which participants did not have access to. But the participants were still allowed to make new submissions byt adjusting their models based on the score they received. At the end of practice generalization phase, the participants were required to select upto three submission id's on AICrowd platform for the submissions they made either during development or practice generalization phase. The final scores were computed by evaluating on these selected submissions and teams were ranked according to the best score achieved.

C Details on Contest Submissions

In this section, we provide details on the approaches submitted by the participants of the top 6 teams in the contest.

C.1 Team `Marlshmallows`

The submitted approach was a set of hard-coded policies designed for each individual substrate. The code[14] for these policies is available online, and the logic for each substrate is described below.

C.1.1 Allelopathic Harvest

The highest priority for focal agents is to navigate to the nearest ripe berry. If there are no ripe berries, the focal agents will change the color of the nearest non-red berries to red. If a green player happens to be within zap range while navigating, the focal agent will zap them. Focal agents avoid the zap ranges of background players during navigation. Two versions of this policy were submitted for the competition: one where focal agents prefer red and one where focal agents prefer green berries.

C.1.2 Clean Up

Focal agents initially navigate to the west wall and align side by side. At timestep 35, a role allocation occurs: the two focal agents closest to the water take on the role of dirt cleaners, while the others become apple harvesters. The dirt cleaners are programmed with specific conditions for task switching. Under one condition, they switch from cleaning dirt to harvesting apples; under another, they revert from apple harvesting back to dirt cleaning. In contrast, the focal agents designated as apple harvesters at the wall are assigned a single task; they exclusively harvest apples without any role change. To switch the goal from cleaning dirt to harvesting apples, the dirt cleaners must either see no dirt for the past 20 timesteps while in water or see at least X other players in the water in the past 20 timesteps. Dirt cleaner 1 has X set to 2, and dirt cleaner 2 has X set to 3. Any focal agent who didn't reach the wall by timestep 35 become additional dirt cleaners with X set to 2. To switch the goal from harvesting apples back to cleaning dirt, the focal agent must see no apples for the past 3 timesteps while in grass. Focal agents avoid stepping into the zap ranges of background players.

[13] https://gitlab.aicrowd.com/aicrowd/challenges/meltingpot-2023/meltingpot-2023-starter-kit

[14] https://github.com/benjamin-swain/meltingpot-2023-solution

C.1.3 Prisoners Dilemma in the Matrix

Focal agents begin by navigating to resources if they are visible, or away from corners if no resources are visible. Focal agents prefer red but will navigate to green resources if there are no red visible. Once focal agents have collected at least 2 red or 2 green resources, they will navigate and interact with the nearest interactable player. If there are no interactable players visible at this point, the focal agent will make a full rotation to assist with spotting interactable players. If there are still no interactable players visible, the focal agent will continue to collect resources (or rotate in place if no resources are visible) until an interactable player is found.

C.1.4 Territory: Rooms

The focal agents begin by running through a predefined sequence of actions (rotate and fire claiming beam). This allows the focal agents to detect which neighboring players are focal or background players; if the neighboring player does not perform the predefined action, it can be assumed to be a background player. Next, the focal agent detects the nearest reachable unclaimed or background-claimed wall and navigates to it; this continues until a background player enters or until timestep 70 when the focal agents begin to destroy walls to reach unclaimed or background-claimed walls in other areas. The focal agents make use of breadth-first search to determine the shortest path to goals while avoiding obstacles like walls and other players. The areas that the background players can zap are considered as additional obstacles. Focal agents also detect when the background players have last zapped to take advantage of the delay before they can zap again. When background players are reachable, the focal agents attempt to navigate to get the background player within zap range while avoiding obstacles. When the focal agent is injured or is in the delay period just after zapping, the focal agent will navigate to the farthest reachable area from all visible background players. Focal agents can typically only see 4 out of the 8 other players at the start of the game, so an additional method was added for focal agents to signal to each other that they are focal as opposed to background players. When two players enter each other's view for the first time, they must fire claiming beam to signal that they are focal. Additionally, if any player is seen firing claiming beam more than 8 times, that player is known to be a background player because claiming beam is only used by focal agents to signal to newly seen players, and there are only 8 other players.

C.2 Team `rookie`

C.2.1 Approach Overview

Each agent's policy is implemented as a concatenation of a ResNet module[31], a GRU layer, and an MLP layer. The MLP layer outputs the Q value of each action. We use a pre-defined prior strategy for each society game to select the best action for each agent.

Regarding mixed motivation considerations[2], we assume each agent will engage in cooperative behavior and then consider their competitive behavior. For cooperative behavior, adopt the popular Centralized Training with Decentralized Execution (CTDE) approach and use the VDN value factorization method[32].

In the **Harvest** and **Clean** environments, competitive behaviors could lead to negative effects because there are no competitors at all. To address this problem, we added some NPC agents during training; these NPC perform some collection activities, which lead to improved performance of the focused agents. Figure 13 shows our overall framework diagram.

C.2.2 Prior Strategies

Allelopathic Harvest A challenging problem in this environment is how to distinguish the focal players from the background players. Since being eliminated will bring huge penalties, the cost of accidentally injuring a friend is very high. However, if you do not attack the marked player, you may be killed by the enemy. So, we considered an interesting strategy. When the berries are uniform in color, the focal player changes its color to an unpopular blue color, thereby distinguishing the focal players from the background players. However, the red and blue serial numbers were reversed during the visualization process. We were not able to discover this problem until the last day of the competition, so we achieved quite poor performance in this environment.

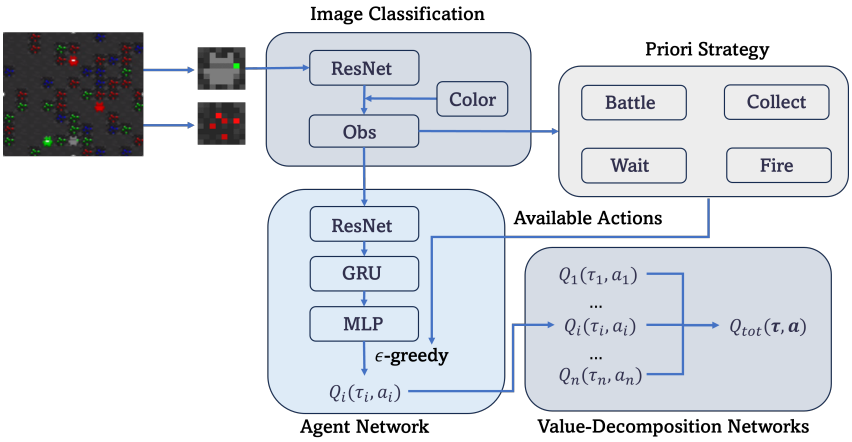


Figure 13: Architecture

Clean Up This is an environment that requires a choice between cleaning and eating. In theory, it is an optimal choice that two players clean the river at the same time. However, due to the presence of some selfish background players, we have to shuttle between the river and the orchard. We give certain penalties to players who stay in open spaces, empty orchards, and clean rivers and add a selfish background player during training to motivate the player's behavior. In this environment, we basically did not adopt any prior strategies.

Prisoners Dilemma in the Matrix This is an environment that is difficult for us to solve. The focus of this environment is still how to distinguish the focal players from the background players. We want the focal players to cooperate with other focal players but weakly cooperate or even strongly defect with other background players. Due to the existence of K-strike explosive background players, the environment encourages us to engage in defection. However, defection among the focal players will bring relatively large losses. Therefore, in this environment, we want the player's strategy to be as pure as possible. When each player is born, its strategy is defined as either cooperation or defection, and it will only collect resources that are the same as its strategy. In order to prevent the focal players from defecting each other, we set the focal players that choose the defect strategy to be unable to actively interact.

Territory: Rooms In our opinion, the players in this environment should be extremely aggressive. Since players cannot be resurrected, one player should be able to occupy a large amount of resources. However, for the focal players, attacking each other can take a huge toll. So, we found a way to differentiate the focal players from the background players. In the beginning, players only color the walls in the middle of the four directions. During this process, we can identify focal players who behave similarly to observations. After completing this operation, we can proceed with the normal activities, trying to encroach on the territory of other players.

C.2.3 Limitations

The neglect of the orientation problem led to limitations in the attack strategy, and we had to rely on prior knowledge to guide the behavior of the agents. In addition, the incorrect use of colors in the visualization configurations led to a waste of time and resources in the Harvest environment, as we could not accurately identify and respond to the opponent's actions. The excessive use of prior strategies sometimes interferes with the output of the GRU model, so we need to utilize 2 to 3 consecutive frames of input to accurately predict the behavior of the adversary. At the same time, we realize that the increase in the number of ResNet layers will take up too many memory resources, so we need to carefully calculate the resource requirements when designing the model. In the training, we did not train against agents, which may also be one of the reasons for the final performance limitation. However, in the environment where competition and cooperation coexist, the task of effectively training agents becomes a complex challenge, and we can only rely on a prior strategies to distinguish between focal players and background players.

C.3 Team `Tess-v1`

C.3.1 Reward shaping solution

In this approach, agents are observed periodically, and the reward model parameters are adjusted according to the difference between the goal and the learned actions.

Implementation

First, the approach uses a PopArt implementation for all the experiments and IMPALA model for visual observation. For policy optimization, it uses PPO with clipping technique with minibatches but it hasn't used multiple epochs almost for all training in order to achieve small steps between updates. For all the substrates, the reward functions were between the range -0.1 and 1. Specifically for prisoners dilemma in the matrix arena substrate, we have an additional fully connected layer in order to feed with the inventory informations of the agents. For all the experiments, discounted return was the same as IMPALA implementation without off-policy correction, because it has been trained purely on-policy method. In particular, we can write n-step return as, $G(t) = R_{t+1} + \gamma R_{t+2} ... + \gamma^n v(S_{(t+n)})$ where γ is the discount factor, R is the reward which is returned by environment, and v is the discounted value approximation which is returned by the model.

Below we outline the substrate specific reward functions:

Allelopathic Harvest Open

Reward function 1:

- +1 for eating any color of berry; +1 for replanting red; +1 for killing an agent which plants any other color than red
- -1 for replanting any other color
- -0.1 for killing an agent which plants red

Reward Function 2

- +1.25 for eating any color of berry
- +0.5 for killing an agent which plants any other color than red
- The other values are same as function 1

The reason why we used two different functions for this substrate is that after training with function 1, we observed the agent gives more value to zap the enemy agents and it switched to learn reward function 2 for the rest of the training. The training has been made with 8 focal agents and 8 random policy agents. We noticed that self-play training is not enough for this kind of reward functions. Because after some training, nobody plants any color other than red and the agent forgets the value of zapping the enemy agents. In the second training, we changed the environment to 12 random policy agents and 4 focal agents in order to be sure 0.5 reward is valuable for the agents. This way it gets lower reward but with higher possibility to get. We used the same agent for green lover agents because the team couldn't find a good way to train two different agents who can collaborate with each other. Instead, we created a really dominant red lover agent. Also for the PopArt implementation, it used three value heads; eating rewards, replanting rewards and zapping rewards.

Clean Up

Reward function:

- +0.1 for eating an apple
- +0.1 for cleaning the lake

For the clean-up substrate, the reward function is a little bit complex than the others. The environment holds the cleaner's data for the last 10 time steps and whenever any agent eats an apple, the cleaners get rewards based on their data. This way cleaners also learn how to clean for increasing the apple production possibility. Spamming the clean action is not enough, it should also learn where to clean. For the PopArt, it has two value heads; eating apple and cleaning the lake.

Prisoners Dilemma In The Matrix Arena

Reward Function:

- +1 for collecting cooperate box
- +30 for interaction with cooperate
- -10 for destroying cooperate task
- -1 for collecting defeat box

We used different gamma values, 0.95 for this substrate. The agents can't play the game for a while after they've been in an interaction with each other. It causes them to not get future rewards which means early rewards are always much more important than the future rewards.

Territory Rooms

Reward Function:

- +0.75 for capturing a wall
- +1 for killing an agent
- -0.05 for destroying a wall

For the territory rooms substrate, we used two different type of environment. The first one is created for team play, where nine focal agents playing with each other. And the second one is created to improve individual play for the exploration of the map,where one agent basically does what he wants to do while the others don't do anything. We used 4 parallel environments and 3 of them were type 2 environment and 1 of them was type 1 environment. This way it can learn capturing the whole map with individual playing and also it learns to be a team player in type 1. Also for environment type 1, it doesn't get any reward for capturing a wall if it is captured by an agent before.

C.4 Team `monkey_king`

This solution builds a rule-based policy for each substrate. The process is divided into two parts: (i) Extract the attributes of each sprite from the RGB observations, that is, parse the specific semantics for each sprite; and (ii) Build a rule-based policy based on the extracted semantics.

Extract the attributes from RGB observations

We parse the attributes of one sprite by identifying the RGB value at some specific positions in the 88 × 88 observation without downsampling. The attributes include background type, whether there is an agent, the direction of the agent, the color of the agent, etc. There may be different attribute types for different substrates, and the attributes of each sprite are represented by a dictionary.

Build substrate-specific rule based policies

In this phase, detailed rule-based policies are designed to be employed for each substrate. While deferring the readers to Appendix for full details, we briefly provide an example of usce rules for each substrate.

In **Allelopathic Harvest**, the agent first checks whether it is currently in the cooldown period for zapping, whether there are green or blue agents that can be zapped, and whether there are no red or gray agents within the beaming range. If these conditions are met, the zap action is executed. If the zap action is not performed, the agent then considers whether there are non-red unripe berries within the planting range. If so, it will plant red berries. If neither of the above actions are performed, the agent will move towards ripe berries. The agent evaluates the four directions based on its role and the current observation, and selects the action with the highest score. The scoring criterion involves multiplying the reward of the ripe berries by a distance-discounted weight to obtain the score of the berry, and then adding the scores of the berries in each direction to determine the score of that direction. In the scenario where there are no ripe berries in the observation, the agent will move to the nearest unripe berries. Note that even if the agent's role is to prefer green berries, it will still plant

red berries. However, green berries will be given a higher score than red berries when choosing the moving directions.

In **Clean Up**, if *Timesteps*< 30, the focal agents first determine their orientation by turning viewpoint based on initialized position, then plan a path based on direction and location to meet the river that runs from the reservoir above to the grass below, forming a line in the order in which they arrive and remaining still. After testing, they can almost successfully reach the designated position in less than 30 steps. *Timesteps = 30.* Since all focal agents using the same strategy have reached the designated rendezvous point and formed a certain order, they begin to assign roles according to the number of their teammates. The allocation scheme is as follows:

- Focal agents number = 2: one “conditionally clean" and another “eat".
- Focal agents number = 3: one “clean" and two “eat".
- Focal agents number ≥ 4: one “clean", one “conditionally clean", the rest “eat".

If an agent on its way to a meeting point is hit by a background agent using a beam and moves out of the game and fails to reach the meeting point within 30 moves to identify a teammate, it is assigned the “conditionally clean" role by default. *Timesteps* > 30. Perform tasks according to their assigned roles. The role of “clean" stays in the sewage pool to clean up, the role of “eat" stays in the apple orchard to eat apples, and the role of “conditionally clean" changes its role according to its observation for a period of time. If there are few pollutants in the field of vision in the previous period of time, it will eat apples. On the other hand, if the number of apples in view is very low a while ago, it will clean up. In *Prisoners Dilemma in the matrix: Arena*, the agent upon rebirth, initially orients itself in a reasonable direction and then proceed towards the center. Upon detecting a desired resource within its observation, the agent will employ depth-first search to identify the shortest obstacle-avoiding path. If it encounters another agent heading towards the same resource, the agent will switch to an alternative resource within its observation. Once a sufficient quantity of resources (set to 5 in our final version) has been collected, the agent will initiate active interaction. It will continue moving towards resources to identify potential interactive partners, as the likelihood of encountering other agents near resources is typically high. After discovering another interactive agent within a certain range of vision, the agent will move towards it. While the agent actively seeks cooperation, it will shift its approach to defection (set to 5 in our final version) if it experiences repeated defections. The agent will collect red resources and interact with other agents thereafter.

In **Territory: Rooms**, if *Timesteps* < 35, the strategy is first to go around agent’s own room and paint all the walls of the room in their own color, without using a claiming beam to paint the opposite room.

Timesteps in [35, 100). The agent stands in the center of the room and constantly looks around to monitor. If it observes someone using the claiming beam to paint the walls of their room another color, it will retaliate by using the claiming beam to paint both its own and the other person’s walls their own color. If the agent observes someone trying to break through a wall to invade its territory, it will go to "high alert", which means walking up to the broken wall and waiting, and once the opponent breaks through the wall and enters his territory, the agent uses the zapping beam to remove opponent from the game. If the opponent does not use the zapping beam again to destroy the wall and enter after waiting for a period of time, the agent will return to the center point and continue to look around the monitoring.

Timesteps in [100, 200). Add some aggressiveness to the agent. If the agent looks around and sees that a wall adjacent to his room is unoccupied, the agent will use the claiming beam to capture it and then return to the center of the room to look around. Surveillance also involves defending against other agents trying to invade their territory.

Timesteps ≥ 200. The aggressiveness of the agent is further strengthened. In the game, standing in the center of the room can see most of the room above itself, so if the agent finds that most of the walls in the room above have not been claimed, the agent will break the walls to invade the room and attack the people in the room. After a successful intrusion, the agent will paint all the surrounding areas of the room in its own color, then return to the center of the room, look around, and if there is a room that satisfies the same condition, it will invade the next time.

C.5 Team `MeltingTeam`

Our general approach for this competition was to implement stable policy optimization algorithms for the individual agents, develop substrate-agnostic learning protocols, and fine-tune the approaches to the individual challenges of the substrates. During the competition, each of us took ownership of one substrate while sharing insights and takeaways with each other. We had a weekly meeting to discuss progress, agree on the next tasks, and divide the workload.

Substrate Agnostic Implementation

We primarily followed the MeltionPot 2.0 whitepaper [2] for the agent's policy optimization, however, we deviated from it at certain points based on our experiences or implementation difficulties. Our codebase is built on the baseline code provided by the organization. We tested the following approaches for the agents' individual policy optimization algorithms.

1. We evaluated PPO [28], A3C, A2C, [29] and Impala [27] implementations in RLlib and decided to work mainly with A2C and Impala based on performance and stability.
2. We augmented A2C and Impala with a PopArt layer [30] that we found valuable to generalize parameters across substrates. It also facilitated a faster and more stable convergence, especially, for the substrates in which rewards could grow rapidly and change orders of magnitudes (e.g. Territory Rooms). We found that lowering the learning rate of the PopArt layer is important to avoid local optima.
3. We implemented a variant of A2C and Impala in which the value network had full observation of the game. We observed a significant decrease in training speed and no clear performance improvement, therefore, decided not to use it further.
4. We tried to implement Contrastive Predictive Coding [35] as well, however, we faced implementation issues and decided to prioritize our efforts on other ideas.
5. We implemented weight sharing on various levels; the whole population, roles (e.g. preferences in the Alleathoric Harvest substrate), and personas (described below).

While vanilla self-play worked well on some substrates, in most cases it lacked stability and generalizability, we addressed these problems with the following approaches.

1. We defined personas for each role, e.g., prosocial and anti-social ones whose reward signals are positive or negative concerning the population's average reward, respectively. Agents randomly chose a persona at the beginning of each episode to follow and optimize.
2. Reward sharing between agents either for the whole population, roles, or personas.
3. We tuned the reward signal for each specific substrate to overcome their respective challenges. Details follow in the next section.
4. We divided training into phases in increasing difficulty and introduced new social aspects in each stage. Mainly used for Territory Rooms and details are described in the respective section.
5. We introduced pre-trained bots from the evaluation scenarios. It did not help significantly while slowing down our training due to implementation difficulties. We did not use this for the final solution.
6. We experimented with sampling different scenarios for each episode during training in an attempt to enhance generalization performance. However, the training process became unstable because of divergent reward ranges, and normalizing them did not resolve the issue. We opted to abandon this approach due to time constraints.

Substrate Specifics

Alleathoric Harvest: The substrate faces the challenge of effectively deciding between planting the personal favorite berry type and planting the globally dominant berry type since it ripens faster. Given that agents possess only partial observations of the environment, identifying the dominant berry type is difficult, and reaching a consensus on the type of berry to plant among agents is equally challenging.

To address this challenge, we implemented a custom reward function. Agents are incentivized to plant a predefined berry (red), which we manually selected. Additionally, agents receive instant rewards for converting other berries to the predefined type. On the other hand, there is a penalty for switching from the predefined berry to other types to prevent the accumulation of rewards through constant back-and-forth changes. Due to time constraints, we were unable to implement a dynamic predefined berry, one that rewards based on the actual global dominating berry.

All agents sharing the same role are trained with a shared-weight policy to enhance sample efficiency. We also fine-tuned rewards for zapping (to encourage more aggressive behaviors) and being zapped. In the end, we achieved optimal performance by not rewarding zapping and penalizing being zapped with a -2 reward.

Prisoner's Dilemma: We believe the ideal strategy for this substrate is cooperating with the members from the focal population while defecting the background population. However, it is intrinsically difficult to distinguish focal agents from background agents only based on image observations. Therefore, we started by training pure cooperating strategy in this challenge, because the cooperating strategy generally provides a higher mean focal outcome in scenarios compared with the purely defective strategy. To do this, we modified the outcome matrix to give a reward of 3 for mutual cooperation and zero for any other outcome. All agents were trained with a shared policy for higher sample efficiency.

Another challenge for this substrate is the lack of resources. The default rewards encouraged taking as many resources as possible. In this case, some agents will obtain all the resources, while others fail to gather any resources to interact with others. We addressed this problem by designing penalties for consuming more than one resource per agent. This further improved the performance according to our evaluations. Due to time limits, we were not able to investigate learning the aforementioned ideal strategy to behave depending on the prediction of the "identity" of the other agents, which could be an interesting future work.

Clean Up: The main challenge in this scenario is the sparse reward feedback. The agent can only receive rewards by eating apples, however, if they do not know how to clean the river efficiently, there will be no apple growing and as a result, no reward feedback. Therefore, we utilize reward shaping to encourage the agents to clean dirty blocks. More concretely, we provide additional rewards for agents, such that, on the one hand, it is proportional to the number of polluted blocks they cleaned to encourage more efficient cleaning, and on the other hand, it is also lower than the reward of tasting apples so they will not interpret that cleaning river is their fundamental goal.

Besides, each agent takes the average reward of all agents as the reward in training, and in this way, they are encouraged to have prosocial behavior. Moreover, we also leverage the symmetricity of agents by sharing the policies for all the agents, and we observe it results in more sample-efficient learning.

Territory Rooms: Our biggest challenge in this substrate was to encourage exploration of the agents, i.e., breaking walls and entering other rooms while learning how to defend themselves. Our final approach was based on a three-step learning procedure as follows.

1. Teach a single agent with others being idle. A larger additional reward was given to claim the resources the first time and a smaller one to reclaim them. The agent learned to explore the whole state space, break walls, and claim resources while ignoring the other agents.
2. Two agents learning concurrently with 7 others being idle. Both agents were initialized from the one in the previous step and they share weights. We provided additional positive rewards for initially claiming resources and zapping others while negative rewards for reclaiming resources, using the beams, and being zapped. We aimed to maintain the incentive to explore while engaging in conflicts and learning how to defend themselves. Penalizing zapping led to fewer resources being destroyed.
3. In the last step, we used the same structure as before but with 3 or 4 agents. We observed marginal improvements only on top of the previous results.

A further observation about this substrate was that PopArt and Impala helped for stability but a smaller learning rate was better for PopArt compared to the values reported in the whitepaper. With the initial learning rate, PopArt scaled the value function too quickly and agents fell into a local optima where they did not leave their own room.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section 1.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 2 and Section 3.
   (b) Did you describe the limitations of your work? [Yes] See Section 3.3 and Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 4.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 2, 3 and 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 3.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 2.5 and Appendix B.3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]