# A Algorithms Pseudo-codes

**Algorithm 1** Online Regulatory Agent Algorithm

**Require:** $\upsilon, \sigma, \mathbf{v}, N$

**for each** $\overline{\mathbf{v}} \in \mathbf{v}$ **do**
  **for** $i = 0,\ \ i \leq N,\ \ i = i + 1$ **do**
    $S \leftarrow getCurrentState(\phi_i)$
    $R'_{inc,i} \leftarrow TPESampler(\overline{\mathbf{v}}_i)$
    $S' \leftarrow modify(S, R_{inc,i})$
    $\omega \leftarrow$ if Omniscient : $\upsilon_i$ else Model-based : $\sigma_i$
    $A_i$ += $simulate(S', \omega)$
  **end for**
  $\overline{\mathbf{v}} \leftarrow |getTargetVector() - A|.mean()$
  $TPESampler \leftarrow ASHA(\overline{\mathbf{v}})$
**end for**
**return** $R_{inc} \leftarrow \mathbf{v}.min()$

**Algorithm 2** Property-based MADDPG. Lines 6-9, in red, represent the critic network $\psi^t$ update process. Lines 11-13, in blue, refer to the true agents' model network $\psi^t$ update.

**Require:** $\Lambda, \upsilon, \psi, \upsilon^t, \psi^t$

1: **for** $i = 0,\ \ i \leq N,\ \ i = i + 1$ **do** ▷ For each agent
2: $\delta \sim \Lambda$ ▷ Generating the mini-batch
3: $s_i, a_i, s'_i, r_i \sim \delta$ ▷ Sampling from mini-batch
4: $\mathbf{Q} \leftarrow Q(s_i, \mathbf{a}_i)$ ▷ Getting Q-value
5:
6: $\mathbf{a}'_i = \upsilon^t_i(s'_i)$ ▷ Generating action
7: $target\mathbf{Q} \leftarrow \psi^t_i(s'_i, \mathbf{a}'_i) * \gamma + r_i$ ▷ Calculating target
8: $\boldsymbol{\ell}_{\sigma_i} = \sum_{j=1}^{|\delta|} \frac{[(\mathbf{Q}^j - target\mathbf{Q}^j)^2]}{|\delta|}$ ▷ Calculating loss
9: Update $\psi_i$ from loss $\boldsymbol{\ell}_{\sigma_i}$
10:
11: $\tilde{\mathbf{a}}_i \leftarrow \upsilon_i(s_i)$ ▷ Generating action
12: $\boldsymbol{\ell}_{\upsilon_i} = -\frac{\sum_{j=1}^{|\delta|} \psi_i(s^j, \tilde{\mathbf{a}}_i)}{|\delta|}$ ▷ Calculating loss
13: Update $\upsilon_i$ from loss $\boldsymbol{\ell}_{\upsilon_i}$
14: **end for**

***Hyper-parameters*** Our time frame is equal to one year. The discount factor is $\gamma = 0.1$. We update the networks to the target networks every 250 steps. Our exploration and exploitation $\epsilon$ coefficient is equal to 0.3. The size of our study vector is $M = 50$. We use a low learning rate value of $1e-4$, to overcome the issue of agents becoming stuck in local minima and struggling to learn new policies.

## B Reward-Shape algorithm

For the implementation of our Reward-shape baseline, we define that the incentive is calculated following the equation:

$$R_{inc,i} = R_{max} - |target_i - \frac{\sum_{j=0}^{M} a_j}{M}|, \tag{1}$$

where $R_{max}$ is the maximum reward delivered by the domain, $target_i$ is the percentage of land that the government defined as a target that agent $\phi_i$ must save in its property for pollinators' sustainability and $\frac{\sum_{j=0}^{M} a_j}{M}$ is the actual percentage of the land saved for the pollinators. Therefore, the actual incentive $R_{inc,i}$ represents the difference between the target and the actual percentage of the land saved for pollinators plus a standard bonus ($R_{max}$).

## C Training Policies

We propose a curriculum learning application to perform our training process. We implement four rule-based policies to define the agents' behaviour under the regulatory agent's influence and model the incentives during the training. It is important to note that the regulatory agent is introduced only after all policies have been executed, providing the correct incentives as a part of the final reward. This strategy can generalise our model by considering each current objective of our regulatory agents.

- **Policy #0:** We train agents without any interference from the regulatory agent or rule-based policies, hence, agents are rewarded purely for their income.
- **Policy #1:** The rewards for agents will be delivered according to randomly generated upper and lower pollination thresholds. When an agent's average pollinator count exceeds the upper threshold, it receives the maximum reward. If the count falls within the defined bounds, it is rewarded with half of the maximum reward. However, if the count is below the lower threshold, the agent receives no reward.
- **Policy #2** It is similar to policy #1, except that all rewards are reversed. The maximum reward is now a negative reward while no reward becomes the maximum reward.
- **Policy #3** to extend the variety and flexibility of policies, the last policy is applied only to neighbouring land cells with other agents: if those agents have a higher percentage of pollinators, they are rewarded for being helpful to their neighbours (sustainability).

## D Problem Description

***Terminology*** Before presenting the model details, we define a set of terms, which will be recurrent in the text:
**(i) decentralised agents**: agents that act and control part of the environment. We refer to them as "decentralised agents", "static agents", "landowners", or simply "agents".

**(ii) regulatory agent**: the agent responsible for distributing incentives to the decentralised agents, referred as "regulatory agents" or "government" throughout the text.
**(iii) property**: a set of cells that a landowner owns. Each agent owns a property and all properties together define the original environment, our "property-based environment".
**(iv) incentives**: actions of the regulatory agent that affect the rewards gained by each agent to promote sustainability. Negative incentives are denominated as "penalties".
**(v) pollinators**: is the environment feature that measures the sustainability of the community. The total amount of pollinators depends on the agents and government actions.

***Formal Definition*** Our model extends *n*-player Partially Observable Stochastic Games (POSGs), mainly due to its capability of enabling the design of decentralised action selection by the agents that compound the problem. A POSG is defined as a tuple $(\mathbf{\Phi}, \mathbf{S}, \mathbf{A}, \mathbf{Z}, T, R, O)$, where $\mathbf{\Phi} = (\phi_1, \phi_2, \cdots, \phi_n)$ is the set of agents acting in the environment, $\mathbf{S}$ is the set of states, $\mathbf{A}$ is the set of actions and and $\mathbf{Z}$ is the observation space for each agent. Moreover, given $\mathbf{A} = (\mathbf{A}_1 \times \mathbf{A}_2 \times \cdots \times \mathbf{A}_n)$, $T$ is the transition function, $T : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$; $R$ is the reward function, $R : \mathbf{S} \times \mathbf{A} \rightarrow \mathbb{R}$; and $O$ defines the observation probabilities, $O : \mathbf{S} \times \mathbf{A} \times \mathbf{Z} \rightarrow [0, 1]$.

We *extend the POSG model* by considering that the environment is partitioned into several cells, each one is defined as $c \in \mathbf{C}$ and is associated with parameters $\mathbf{p} \in \mathbf{P}$, with $\mathbf{P}$ being the set of all possible parameters. Each agent $\phi_i$ controls a subset $\mathbf{C}_i$ of cells (property), where $\mathbf{C}_i \subset \mathbf{C}$, $\bigcup_{\phi_i \in \mathbf{\Phi}} \mathbf{C}_i = \mathbf{C}$, $\bigcap_{\phi_i \in \mathbf{\Phi}} \mathbf{C}_i = \emptyset$. Moreover, each property $\mathbf{C}_i$ is associated with a vector of parameters $\mathbf{p}_i$ (with each parameter value $p$ related to a cell $c$), that represents the outcome from the decisions taken by the agents.

The reward function is defined as $R_i = \alpha_i \times R_{local,i} + (1 - \alpha_i) * R_{global} + \frac{R_{inc,i}}{2}$; where each agent has $\alpha_i \in$ [0,1]. $R_{local,i}$ is a local reward for each agent $\phi_i$, defined as $\sum_{c \in \mathbf{C}_i} F(c)$, where $F$ is a reward function for cells $c$ and is defined according to the domain. $R_{global}$ is a global reward defined as $F_g(\mathbf{C})$, where $F_g$ is a function that gives a reward to all agents based on the global state of the environment. $R_{inc,i}$ is an incentive reward for each agent based on its alignment to the government's goal, decided by the regulatory agent. The regulatory agent has its own reward function $R_{reg}$, which is tightly coupled with agents' alignment to expected goals and regards the complete current state of all cells $\mathbf{C}$. $R_{reg}$ is modelled according to the target domain. We present our function in Section **??**.

For each agent $\phi_i$, an action in $\mathbf{A}_i$ is a vector $\mathbf{a}_i = [a_1, a_2, \cdots, a_M]$, where $a_j$ changes the parameters of a cell $c_j$, and $M = |\mathbf{C}_i|$. On the other hand, for the regulatory agent, an action is defined as the vector $[R_{inc,1}, R_{inc,2}, \cdots, R_{inc,n}]$, where $R_{inc,i}$ affects the reward of each agent $\phi_i$. However, note that since $R_{inc,i}$ is associated with an agent and, hence, to its property, each $R_{inc,i}$ is a vector of $r_{inc,i,m}, m = 0, 1, ..., M$, which describes the incentive for each cell in the property of $\phi_i$. Each agent takes decisions independently, but they suffer indirect influence from the environment (modified by others). Also, each agent $\phi_i$ aims to maximise its expected cumulative discounted reward $V_i = [\sum_{t=0}^{\tau} \gamma_n^t \times R_i]$, where $\gamma \in$ [0, 1) is the discount factor and $\tau$ is the time horizon of the coordination problem. The regulatory agent, on the other hand, aims to bring the environment to a state $s^*$, which maximises its reward $R_{reg}$ (based on the parameters of all cells $\mathbf{C}$). To do so, it must decide on incentives to give to the other agents to achieve its objective.

***Environment Representation*** As aforementioned, our environment is described as a set of cells $\mathbf{C}$ with parameters $\mathbf{p}_i, \forall c_i \in \mathbf{C}$. Practically, we model it as a 2D matrix with a size equal to the number of cells in our environment $|\mathbf{C}|$, which stores each parameter $\mathbf{p}$ in its respective cell $c \in \mathbf{C}$ – reffered as the *Environment Matrix*. This representation is able to provide us with all the information needed to perform planning. However, while training the decentralised agents (in real-world and also the model of each agent inside ORAA), we need to extract more precise information to learn each feature separately in the environment, considering each agent and its properties individually. To do so, we extract 3 new matrices from the original one:
**(i) Current State Matrix** that represents the current status of each agent's calls (lands), i.e., represents the land $\mathbf{C}_i$ with parameters $\mathbf{p}_i$, for the agent $\phi_i$.
**(ii) Incentive Matrix** that represents the regulatory agent action in the environment for the agent $\phi_i$, i.e., the vector $R_{inc,i}$ with the incentives for each cell in $\mathbf{C}_i$, and;
**(iii) Action Matrix** or **Pollination Matrix**, in our domain, indicates pollinators each agent wishes to have on the land (i.e., represents the actions $\mathbf{A}_i$ taken by $\phi_i$).

Figure 1 illustrates how the regulatory and decentralised agents see the environment and translate it as inputs for training. In the example, two decentralised agents share the environment (blue and yellow). Each agent has its own property. Red numbers highlight the cells to which the agent does not have access, and the green ones are their property. The regulatory agent can access the information of both agents. Considering this model, the planning process starts.

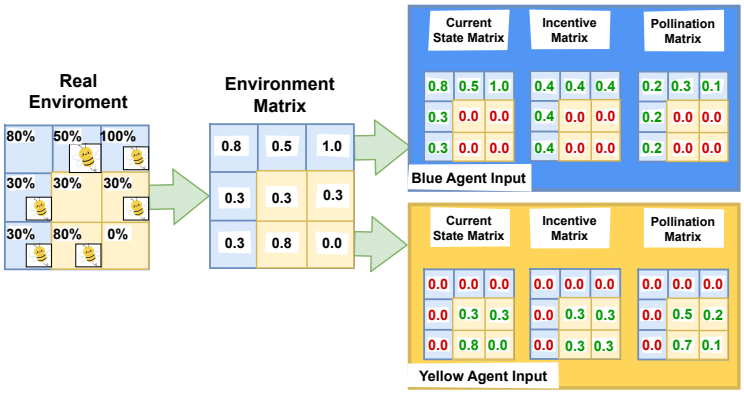


Fig. 1: Real state transformation into inputs for our model.

## E Pollinators Game

We propose the *Pollinators Game* to test and evaluate our proposed system of incentives. Figure 2 illustrates the game.

The environment is divided into properties, represented by different colours in Figure 2. Each property can host habitats for pollinators. Agents' actions consider increas-

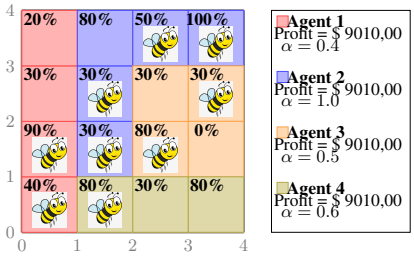


Fig. 2: Pollinator's Game spatial positions and properties of the land cells. Cells with "bees" represent lands that are being pollinated. The numbers at the top-left corner of each cell represent the percentage of the cell with pollinators' nests.

ing or decreasing the habitat size within a cell, using the remaining land for crops, which generate income. However, the actual number of pollinators in a cell is determined by nature, not solely by agents' actions. Pollinators can spread to other cells, with the chance of acquiring new pollinators depending on the distance to and number of neighboring habitats. For instance, in Figure E, a cell with 0% habitat size means maximum crop potential, but without pollination, no crops are produced. To improve this, the agent can increase habitat size or rely on natural pollinator dispersal, simulating the diminishing returns seen in reality.

Suppose an agent with 0% pollinators on its land declares 30% of it for pollination. In the next iterations, it may only see 20% of the land being used for habitat and 20% of the pollinators within the habitat contributing to the chance of pollination. If the agent chooses to maintain the 30% declaration, there is a chance of gaining more pollinators in the habitat. Conversely, reducing the habitat size to 0% will have an immediate effect in the next iteration, removing all habitats and pollinators from the land.

Players are rewarded twofold: **(i)** for crops sold by the end of the turn (local reward) and **(ii)** for the total amount of habitat (global reward), which enables nearby cells to pollinate. Initially, pollinators start existing randomly, and agents must be mindful not to let them go extinct. Each agent has an $\alpha$ parameter that decides how much they value local and global rewards. Additionally, we designed a simple economic system to drive survival and further growth. Each land unit costs money to maintain, so agents are assigned a certain amount of Gold Coins to survive the first few iterations of the game. After that, each land unit has a fixed upkeep cost taken each turn.

The regulatory agent's objective is to ensure the longevity of pollinators and steer the agent towards societal well-being. The target is the difference $\Delta_\phi = |target_\phi - avg\_pol_\phi|$, where $avg\_pol_\phi$ is the arithmetic mean of pollinator habitat within the specific agent's controlled lands, and $target_\phi$ is the desired mean. The closer $\Delta_\phi$ is to 0, the greater the government's success. This defines the regulatory agent's main objective function $R_{reg}$.

## F Our Baselines

We defined two baselines to our study:

**(i)** a simple **Reward-shape algorithm**, which also adds incentives by changing the reward values as we do using the local and global rewards. However, in this baseline, the closer the landowner policy is to the defined target, the higher the reward will be, leading the agents to act sustainably. See the Technical Appendix for further details about the function used to shape rewards as incentives.
**(ii)** the literature algorithm, **LToS** [**?** ]. Also mentioned in our Related Works, LToS is a hierarchically decentralised learning framework capable of incentives agents to act cooperatively in a networked MARL domain which, similarly to us, works in two levels of learning: the regulatory and the decentralised agents levels. We provide in our results section a detailed analysis comparing LToS to our approaches.