Complex Social Systems: Modeling Agents, Learning, and Games
851-0101-86L, Fall 2021

Coding Project: Crop Wars

Christopher Golling, Gael Mourouga, Aaron Moser, Otto Schmidt, Amirhassan Keshavarzzadeh

1 Introduction

Agricultural structures are shaped by a variety of factors, including economic, environmental, cultural, technological and geographical conditions [1].
Consequently, both farmers and policymakers can benefit from farm-level models which optimise water usage and crop nature, based on a set of different externalities including water resources availability and market dynamics.
The model featured in this coding project aims to illustrate the process behind the development of agent-based farm-level models, by starting with a simple, deterministic model that is progressively complexified with elements from physics-based models, game theory and reinforcement learning.

2 State-of-the-Art

To guide the development of our illustrative model, a literature review was conducted to assess the state-of-the-art on agent-based modelling for water resources allocation and farming simulations.
Our starting point was the thesis *"Agricultural policies and farm structures: Agent-based modelling and application to EU-policy reform"* by Happe (2004) [1], which outlines in details the development of an agent-based model (AgriPoliS) to assess the influence of agricultural policies at the farm level, which is applied to a case study in the region of Holenhole in Germany.
To have an overview of more recent approaches. a seed review paper was selected, *"A review of Agent Based Modeling for agricultural policy evaluation"* by Kremmydas et al.(2018) [2] from which a literature graph was generated using the online tools Connected Papers, and is analysed on figure 1.
For completeness, the graphs generated through other seed papers were also analysed, including an older paper applying game theory to decision making in farmer cooperatives by Staatz (1983) [3] and a case application of agent-based models to water allocation on the transboundary Nile river by Ding (2016) [4].
One thing which became apparent through our literature review was that the parametrisation of the models with real-life data, calibration and sensitivity analysis, as well as the model validation through the analysis of existing agricultural systems was the most time and ressource-consuming part of the studies. As such, we decided to procedurally generate data related to externalities (crop price and availability, weather data, water resources availability) and to generate a virtual geography for our agent-based model, centering our study around the construction of the model itself and the analysis of the interactions between agents for a given set of externalities.

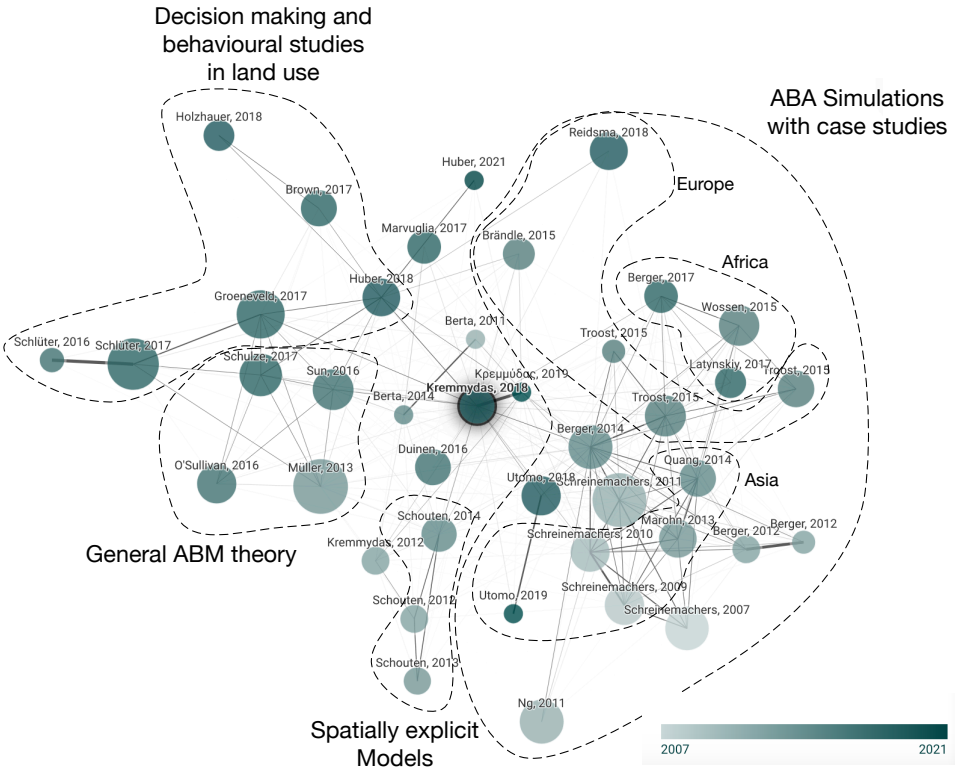


Figure 1: Graph showing papers related to ref. [2] generated on Connected Papers. Circle radius is proportional to number of citation, color to publication date, dashed line refers to paper categories. ABM: agent-based modelling, ABA: agent-based agricultural simulations.

3 The Crop War model

3.1 Concept

The Crop War model aims to visualise how farmers can compete with respect to water resources and crop profitability, given a geographical environment (procedurally generated) and a set of externalities including market dynamics (supply and demand impacting the price of crops), agent personalities and weather conditions.

3.2 Overview

In order to illustrate the development of our Crop Wars model, we organised the model in versions, starting from purely deterministic models with added layers of complexity, on top of which we implemented Reinforcement Learning (RL) to identify the best strategies for a given set of externalities and a specific reward function.
In `v1.1` , the model is not spatially resolved, in the sense that each agent has a single cell (farm) on which it can grow a single crop (out of four different crops) and at each time step decides to sell or stock its yield.
In `v1.2` we implement the `Map` class and the possibility for spatial expansion, where agents can buy neighbouring cells in order to grow more crops.
In `v1.3` , we implement a `Market` model, where the total amount of a given produced crop impacts its price, given a certain demand.
The market also incorporates agent personalities, where different personalities may have

different tendencies to expand spatially or sell and/or stock commodities.
These deterministic versions pave the way to the Reinforcement Learning (RL) model, which is implemented in `v2` . In `v2.1` , a RL agent will play against an expanding agent (i.e. `Introvert`, see section 3.4.4). The reward function, which the RL agent tries to maximize, is the budget evolution.
In the next version `v2.2` , the RL agent will compete against the `Trader` agent (see sec. 3.4.4). In this version, both spatial expansion and market decision play a role.

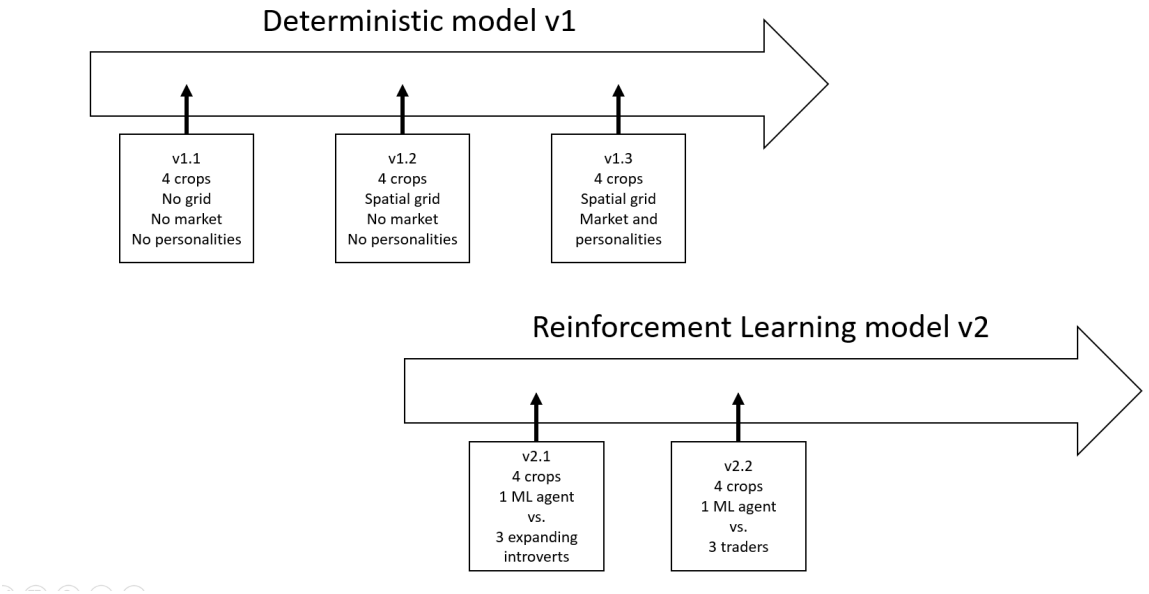


Figure 2: Overview of the different model versions

3.3 Software

In order to build the model, we decided to use the python library `agentpy` which conveniently introduces the `Agent`, `Grid`, `Model`, and `Experiment` classes, allowing to create our model with minimal prerequisite work.
A documentation of the CropWar model was created, to define functions and variables, and make the code more readable. Verbatim text indicates used function or classes. Exact and extensive descriptions of these objects can then be found in this documentation.
All relevant programs and files were hosted on a GitHub repository, with which our team interacted using Visual Studio Code and GitHub Desktop to navigate between different model versions (hosted on different branches of the GitHub tree).
The report and presentation were generated via LaTeX and hosted on Overleaf.

3.4 Deterministic model development

3.4.1 Crop factors

In the current model, four different crops (winter wheat, barley, maize, beans) have been considered for agents to choose from. The property of each crop can be defined by four values: seed cost, sell price, maximum harvest yield and water consumption. Seed costs and sell prices are factors which determine the total revenue [**?**].
Maximum harvest yield is an important factor that is used to calculate the actual crop's harvest yield. The maximum harvest yield is based on empirical data under different climate conditions. For winter wheat, barley, maize, and beans the maximum harvest yields are obtained from [**?**]. Water consumption illustrates how each stage of crop development will decrease under the water stress situation [**?**].
These factors were initialized for the model in `crop` module.
The crops will be numbered throughout the report as follows:

Crop	Crop ID
Winter wheat	1
Barley	2
Maize	3
Beans	4

These coefficients will be used in the FAO model [**?**] as follows :

$$1 - \frac{Y_t}{Y^m} = \sum_{i=1}^{n} k_{y,i} \cdot (1 - \frac{d_{i,t}}{ET_{m,i}}) \tag{1}$$

$$ET_{m.i} = k_{c.i} \cdot ET_o \tag{2}$$

Here Y_t is the crop harvest yield in tons per ha per time step, Y^m is the maximal crop harvest yield in tons per ha, $k_{y,i}$ are empirical non-dimensional values relating water stress to crop yield. The ET_o is the reference evapotranspiration, it considers the effects of the climate on the crop evapotranspiration and is set to 1.0 to reduce the modelling effort. Finally $d_{i,t}$ is the water availability in each time step and $k_{c,i}$ is the crop factor.
The formula (1) is used in order to calculate the harvest yield for each crop in the following model.

3.4.2 Version 1.1

The first version of the model `v1.1` is not spatially resolved, in the sense that each agent can only grow one crop at a time and can not expand to other locations.
The price of each crop is fixed and does not vary as a function of time, and the agents only have the choice between selling or stocking their yields, their storage space being unlimited. Fig.(3) show the evolution of the budget (in $) and stock as a function of time over 10 time steps for 5 agents choosing between 2 crops to grow:

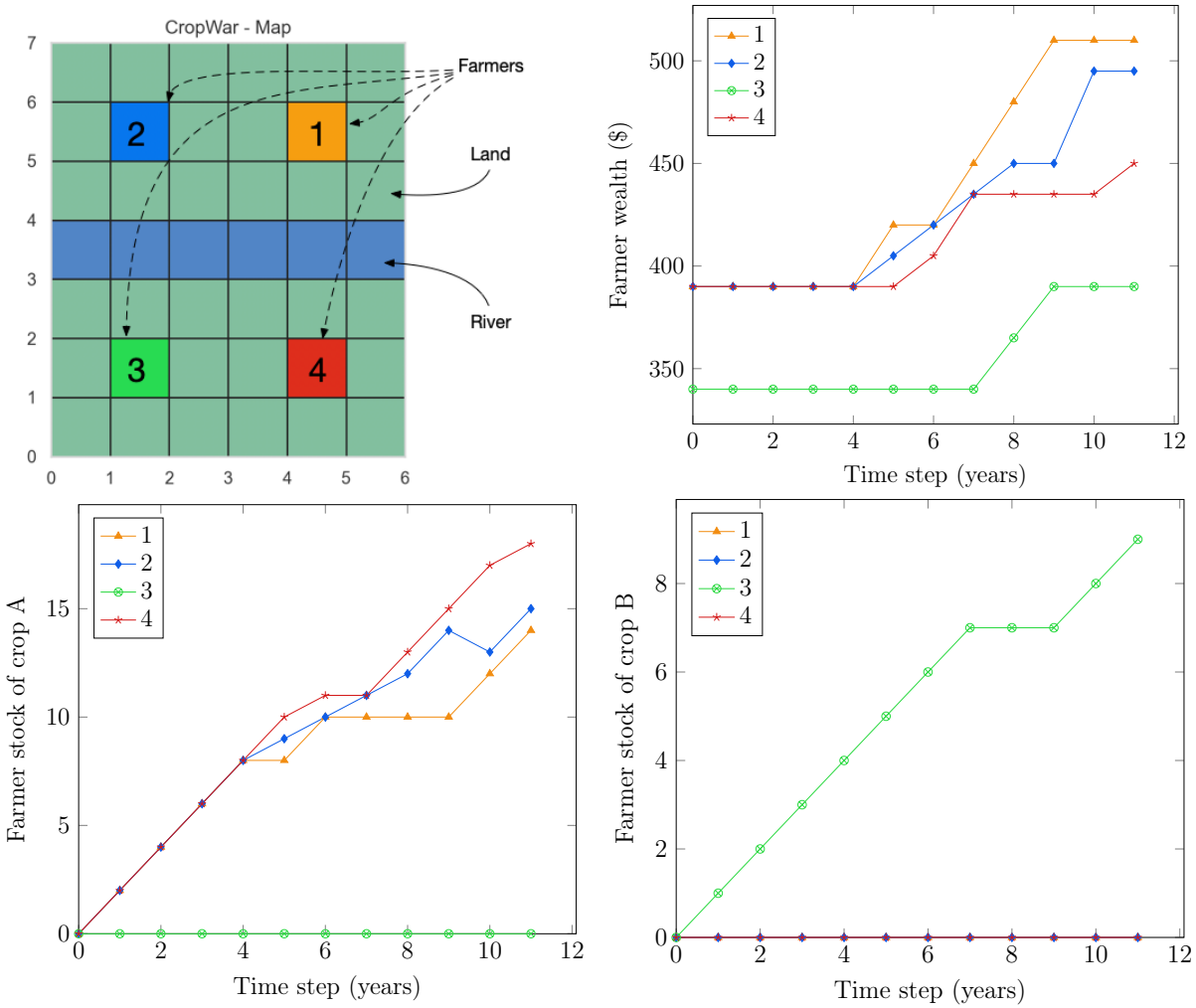


Figure 3: Graph showing the map with four farmers (denoted 1,2,3 and 4), the evolution of crop stocks the and farmer's budget as a function of time over 10 time steps

3.4.3 Version 1.2

In the second version of the model `v1.2` , `Map` class was created, so that agents can expand their farms in order to grow more crops. A `buy_cell_threshold` between 0 and 1 is defined for each agent, representing the tendency of agents to expand their farms (0 corresponding to agents expanding as much as possible, and 1 to agents remaining on their initial cell).

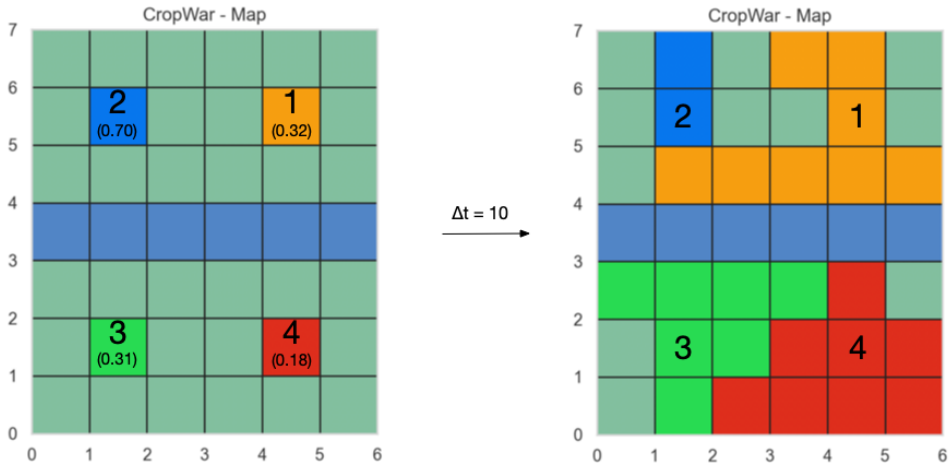


Figure 4: Evolution of farmers' land count on the grid over 10 time steps. Number in brackets indicate the farmer's `buy_cell_threshold`

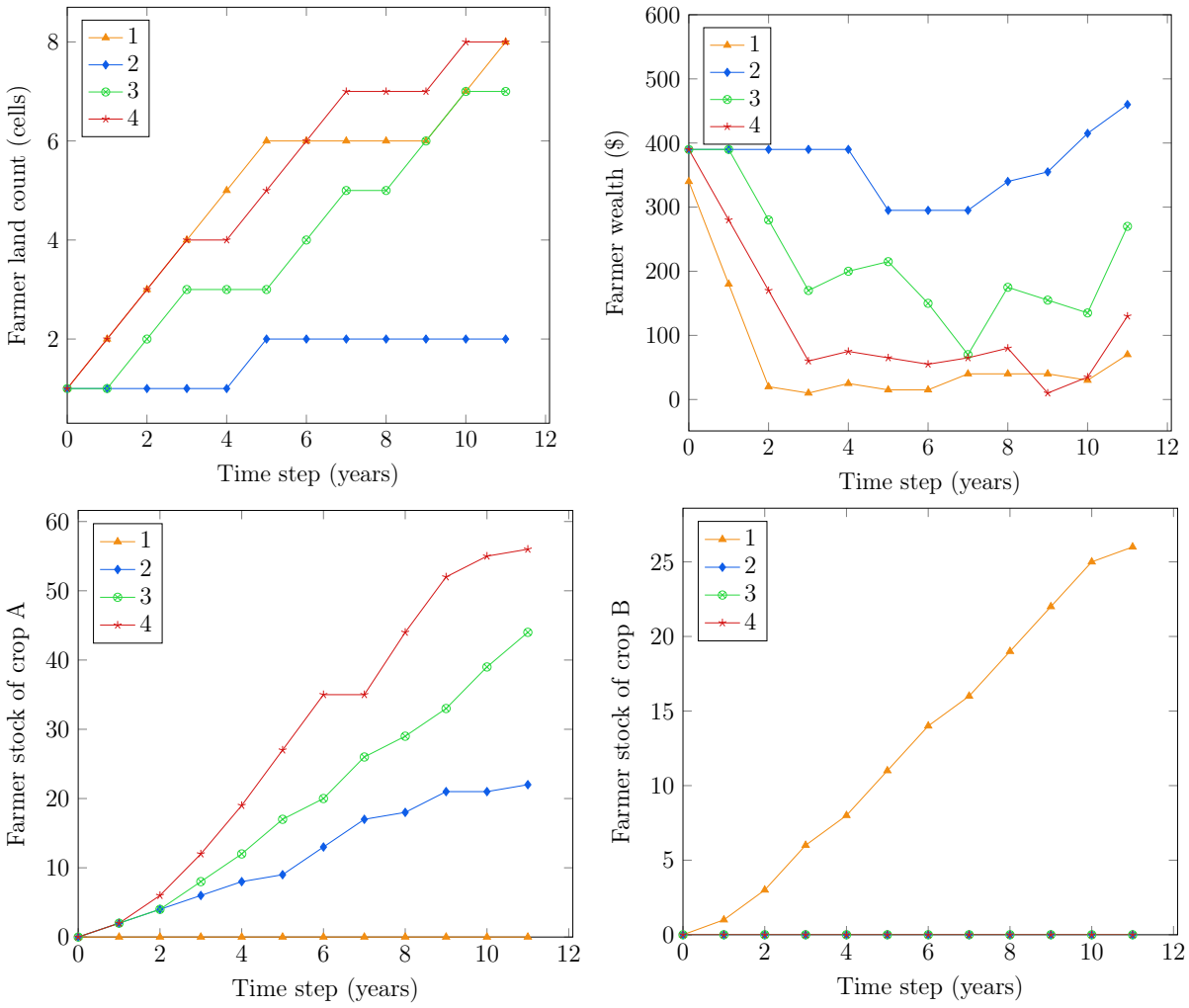


Figure 5: Graph showing the farmer's land count, the evolution of crop stocks and the farmer's budget as a function of time over 10 time steps

3.4.4 Version 1.3

Once the fundamental properties of the model and the `Map` class were defined , the second interaction platform, i.e. the `Market` model, was implemented in v1.3 . In this model, prices aggregate globally.

Commodity demand From [5] in agriculture the demand is usually a strong function of the population to feed, and a weak function of the price.
We therefore define the global demand for crop j as follows:

$$D_j(p_j) = \left(D_0 + at^2\right) e^{-\alpha(p_j - p_{j0})} \tag{3}$$

Where D_0 is a base demand, at^2 is a term representing population (and demand) growth as a function of time t, p_j is the price of crop j at time t, $p_{j,0}$ is the base price of crop j, and α is a demand slope, representing the fact that less and less people will be able to afford expensive crops.

Commodity demand The supply of crop j by farmer i is limited by his stock of this crop s_j, plus the yield at time t Y_j, but more importantly is a function of the willingness of the agent to supply a crop at a given price:

$$S_i(p_j) = A + c_i \cdot p_j \tag{4}$$

Here c_i is the willingness to supply crop j at a given price p_j by farmer i and A is the surplus or his willingness to supply crops for free.

Price aggregation Price of crop j from supply and demand:

$$p_j = p_{j,0} \cdot \frac{D_j(p_j)}{K \cdot \sum_i s_j} \tag{5}$$

Where $p_{j,0}$ is the base price of commodity j, $D_j(p_j)$ the demand and $\sum_i s_j$ the total stock available of crop j across all farmers. The constant K represents a residual stock of a commodity and is equal for all j. This constant is implemented for two reasons:

1. In a realistic market with a sufficiently large supply side, the global stock of a commodity will not reach zero. We assume that there will always be some residual "reserve" stock.

2. If the global stock for a commodity should reach zero, we are confronted with a mathematical singularity. The constant K was incorporated to avoid this inconvenient behaviour.

Agent strategies For now, the performance of all agents only depends on the initial random choice of a position. However, in order to see different outcomes, a set of strategies needs to be defined.
A strategy is defined on an interaction platform. In this model we use two platforms: the market and geographical expansion.
The agents on such platforms can be split into interactive and non-interactive agents (see `agents`), i.e. an agent class able to adapt and a baseline class.
For the market platform the strategy types are:

1. `Trader`: compares prices of market commodities, i.e. crops, and decides to switch to crop j if the expected profit P_j is bigger than the profit for a crop i before $P_{old,i}$:

$$P_j(p_j) = (D_j - s_j) \cdot p_j > P_{old,i} \tag{6}$$

2. `Introvert`: baseline agent. No adaption to market prices. The introvert stays with the initial crop choice and supplies a fixed fraction of the stock.

The strategies of the expansion platform are not discrete and defined via specific agents classes. The tendency to expand is given with the fixed value `buy_cell_threshold`. Each iteration, the agent decides randomly a value $\in [0, 1]$. If `buy_cell_threshold` is bigger than this random value, expansion will not take place. Any value between 0 and 1 however makes expansion possible.

Dynamics Each iteration of this version can be separated into three steps:

1. Harvesting period: agents harvest according to their crop choice and add the harvest yield to the stock.

2. Interaction period: market interaction takes place.

3. Strategy period: the trader type of the farmers will react on market prices and eventually change the crop. Agents can also buy cells (expand) if `buy_cell_threshold` is large enough.

An illustration of this process is given in the graphs below. These show the dynamics of two `Trader` and two `Introvert` agents over 50 time steps.
It is important to note that the different strategies lead to different outcomes. Graph **??** shows the budget evolution of the four agents. Here agent 1 and 2 are `Trader` agents, agent 3 and 4 are `Introverts`. We can see that over 50 time steps, the `Trader` class generates more profits. This can be expected since they are able to adapt to market prices and therefore have an advantage.

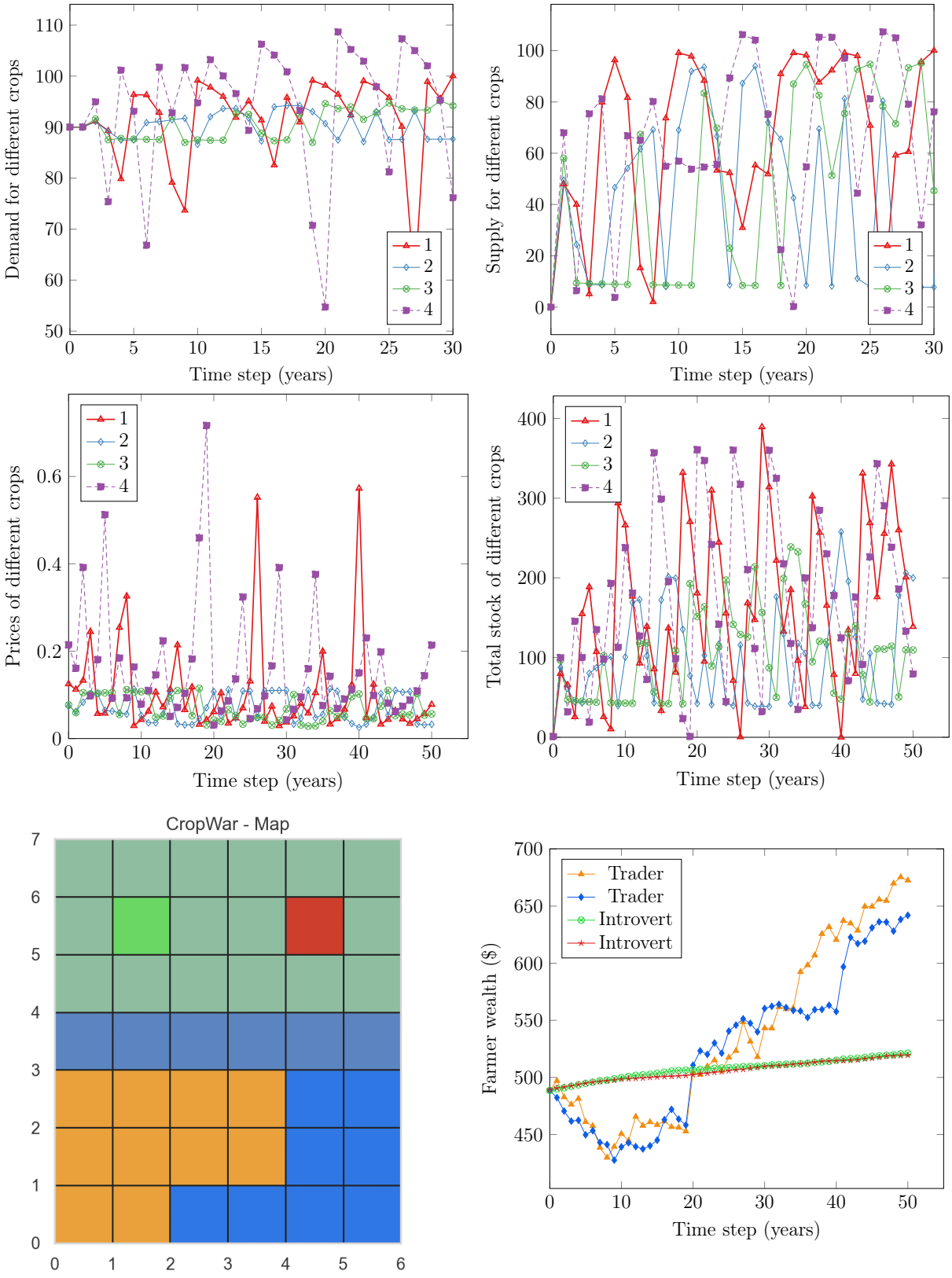


Figure 6: Graphs showing the evolution of supply, demand, prices and global stocks for each crop, with the map and the farmer's budget over 50 time steps using the `Market` model

3.5 Reinforcement Learning model

The preceding versions v1.1 - v1.3 defined the deterministic agents and the model of the environment in which actions take place. In order to see unpredictable, emerging behaviour, the agents would need to adjust to evolutions of the market and learn. This learning process is modelled via a Reinforcement Learning (RL) as explained in this section.
Before outlining the technical implementation of the RL approach, a short introduction to RL will be given.

3.5.1 Reinforcement Learning

Within the vast area of machine learning (ML), reinforcement learning describes learning paradigms that focus on optimal decision-making of intelligent agents.
The decision in RL are modelled with Markov chain processes (MC). The main constituents are the agents, an environment in which the actions take place, states of the agents and an interpreter which determines and interprets the actions of each agent. A schematic of reinforced learning is given in figure 7.

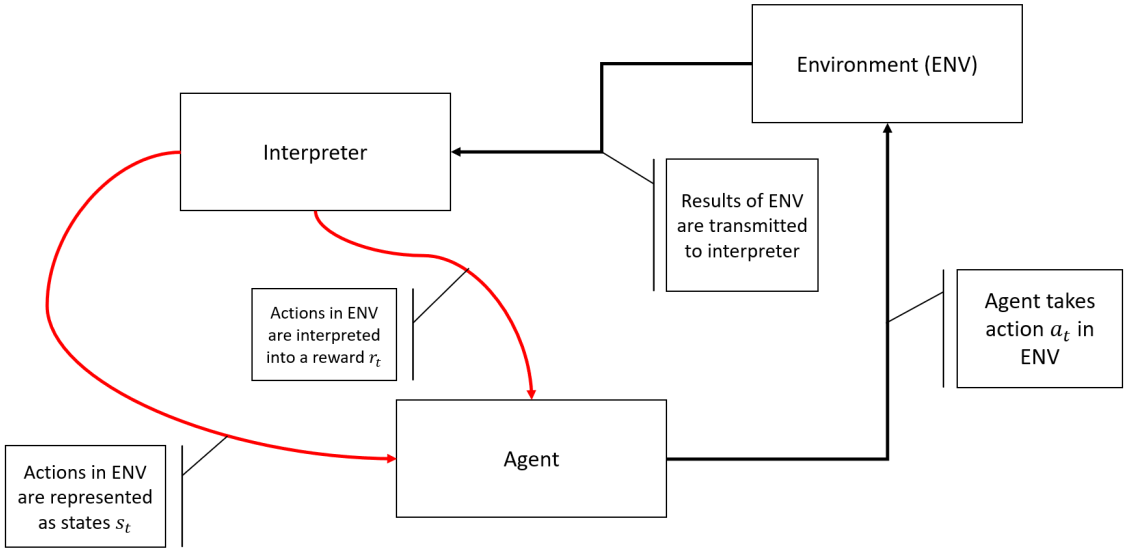


Figure 7: Feedback loop in RL. An agent decides to do a certain action a_t based on his observation of the environment. This affects the environment, which yields a new state. The effect of the agents' action is then interpreted to update the strategy of the agent.

The interactions in the above illustrated feedback loop are conducted in discrete iterations or time steps t. In each iteration, an agent finds itself in a state s_t and with a reward r_t. An action a_t is taken in the environment from a set of possible actions. The actions lead to results, which are then translated by an interpreter into a new state and a reward. In this process, a RL agent works out a policy $\pi(a_t|s_t)$ after which it will act in a specific environment. This policy is chosen such that the cumulative reward over the iterations is maximal.
The agent therefore associates certain rewards to a set of actions in combination with their effects at a . In that sense, the agents act according to their associative memory.

3.5.2 Proximal Policy Optimization

There are various ways to implement the schematic RL process of fig. 7. In this project, we chose the Proximal Policy Optimization (PPO) algorithm. This is mainly because PPO is suited for environments with discrete time steps, continuous observation space and discrete action space as in our model. The theoretical foundations are given in [**?**]. For the software implementation, we follow [**?**].

Policy gradient methods The PPO algorithm is based on policy gradient methods. Here, the main objective is to compute the gradient of a policy objective and optimize it. The policy objective used in [**?**] reads

$$L(\theta) = \hat{\mathbf{E}}_t[log(\pi_\theta(a_t|s_t))\hat{A}_t] \tag{7}$$

where:

1. $L(\theta)$ - Policy loss function.
2. $\pi_\theta(a_t|s_t)$ - The stochastic policy in parameter regime θ given as transition probabilities in the MC of taking action a_t in state s_t. It is a neural network, that suggests an action for a given state based on previous training experience.
3. $\hat{A}_t$ - Estimator of the advantage of the current action. Computed as a discounted reward. This neural net is updated with the experience (i.e. reward) that the agent collects in an environment.
4. $\hat{\mathbf{E}}_t$ - Expectation for current iteration t.

The above function incorporates one main dynamic: if an action a_t resulted in better than average reward during training, the advantage $\hat{A}_t$ will be positive. This yields a positive policy gradient, being defined as the gradient w.r.t. the parameter set θ (see [**?**]):

$$\hat{g} = \hat{\mathbf{E}}_t[\nabla_\theta log(\pi_\theta(a_t|s_t))\hat{A}_t] \tag{8}$$

This would increase the probability of taking the same action decision in the same or similar state. Vice versa, the gradient is negative if the advantage is negative, which reduces the probability of taking the corresponding action.
The above defined method is the core technique in PPO of updating policies. However, there are some subtleties which require adjustments. One of which is the Trusted Region Policy Optimization (TRPO) defined in [**?**], where unwanted effects of self-reinforced action trajectories can be reduced.

Implementation For the implementation of a PPO model, we used the module `Stable-Baselines3` (SB3). Its GitHub repository with all necessary sample functions can be found here. The documentation of the PPO algorithm in this framework can be found here.
As suggested in the documentation of SB3, training was initially performed using the default hyperparameters and the `MlpPolicy` of SB3. By using `TensorBoard` to track the learning, we could visually check the agent's performance with the evolution of the mean reward. If no asymptote was reached, the number of steps were increased. By tweaking the learning rate over three orders of magnitudes, we found the default one of 0.0003 to be stable. The number of parallel environments was increased to eight, in order to get a diverse set of initial seeds. This produced high rewards and interesting behaviour.

3.5.3 RL model versions

As illustrated in fig. 2, we implemented two scenarios. Both differ in the competing benchmark agents. The environment in which the RL agent is trained is defined in section 3.4.4. The RL agent was trained in this environment in 10000 iterations. The reward function that was used for training is given by

$$\frac{1}{3}(b^2 + r^2 + s^2) \tag{9}$$

where b is the fraction of the global budget that the agent possesses, r is the current ranking (1 for first, 0.66 for second, 0.33 for third, and 0.0 for last) and s is the fraction of the global supply that the agent supplies. It was chosen so to maximize profit of each agent.

Version 2.1 This version employs the following setup:

1. Four crops
2. Four agents: One ML agent vs. three `Introvert` agents

The evolution of the budget (i.e. the reward relevant quantity) in this setup is displayed in figure 8. The trained RL agent generates a higher wealth from the first iteration on.

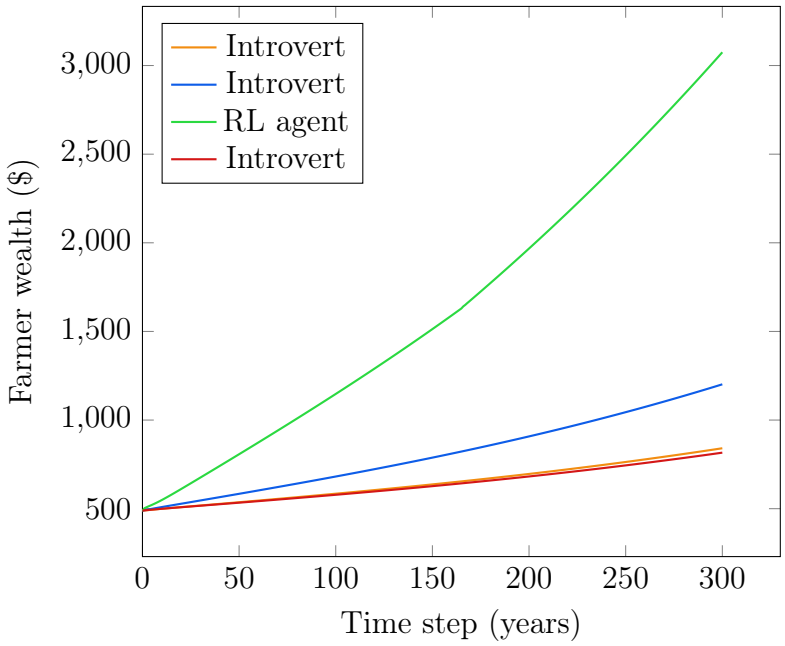
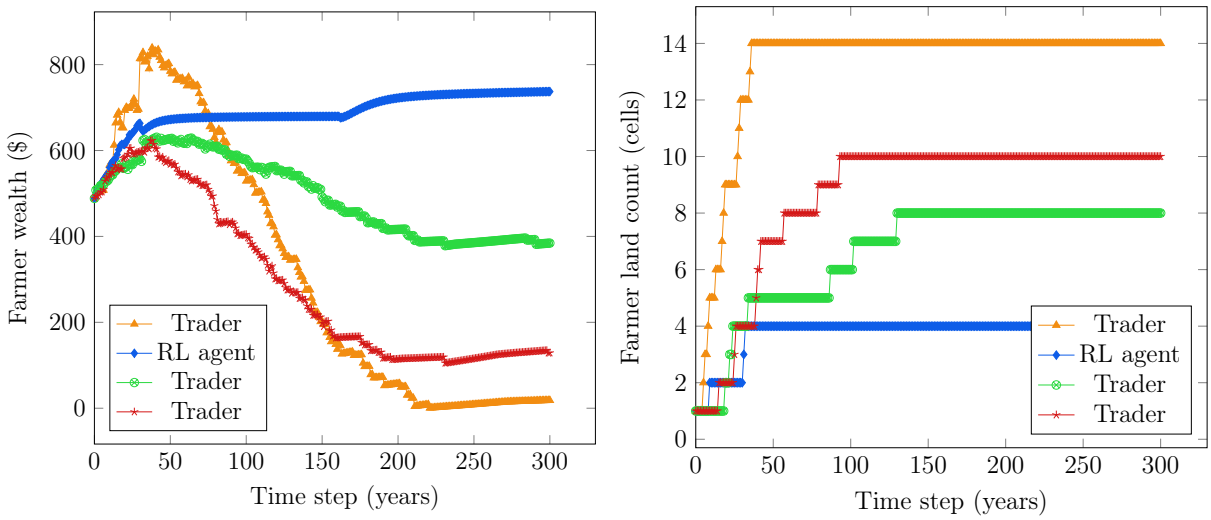


Figure 8: Budget evolution over 300 iterations

Version 2.2 This version employs the following setup:

1. Four crops
2. Four agents: One ML agent vs. three `Trader` agents

The three `Trader` types compete against an RL agent who applies a policy which has been advantageous in 100000 iterations. The budget and cellcount evolution in this setup over 300 iterations is illustrated below.



(a) Budget evolution over 300 iterations

(b) Cellcount of the agents over 300 iterations

Figure 9: Budget and cellcount evolution

The budget evolution in fig. 9a shows how the RL agent generates a higher wealth over 300 iterations compared to the three `Trader` types as benchmark. Interestingly, from fig. 9b we can see that the winning strategy does not lead to a high cellcount since the RL agent has the lowest number of cells after all iterations.
A more thorough insight into the dynamics in this setup can be obtained by observing how each agents stock evolves. This can be seen in figure 10 below. In contrast to the other agents, the RL agents (second plot from left) seem to have a fundamentally different strategy.

Figure 10: Stock evolution for the three `Trader` agents and the RL agent.

3.6 Outlook

The current state of this project allows the pursuit of many directions. Especially with regards to machine learning a multitude of different scenarios could be simulated.
One example is a possible drought scenario. The current code can be easily extended to simulate time-varying water availability scenarios for the river. Thus it would be possible to train the agents to meet a certain demand threshold in each time step while the water availability is reduced. Such a scenario might yield interesting results where the agents have to cooperate, since there is a strong first-mover advantage for farmers that are further up the stream of the river.
Other directions would be to include more realistic agricultural models and processes in order to evaluate agricultural policies, implement more realistic agent personalities, refine the market model, apply the model to other markets with different commodities. Moreover, a game-theoretic analysis of our model might give another interesting perspective. Some of these features are partially or fully implemented already, but could not be adequately tested due to the lack of time.
Since this is our first project with machine learning, the development of the environment and algorithms took relatively long. Hyperparameter tuning of the PPO model would most likely improve the performance of the ML model a lot. Also, maybe there are other, better performing RL algorithms than PPO. For example, we found, that random sampling of the action space outperforms the Introverts almost always. Thus, maybe an algorithm with more drastic changes than PPO, which is designed to only alter the current policy relatively slowly, might be better performing.

References

[1] K. Happe, ed., *Agricultural policies and farm structures - Agent-based modelling and application to EU-policy reform.* Studies on the Agricultural and Food Sector in Central and Eastern Europe, Volume 30, 2004.

[2] D. Kremmydas, I. N. Athanasiadis, and S. Rozakis, "A review of Agent Based Modeling for agricultural policy evaluation," *Agricultural Systems*, vol. 164, pp. 95–106, July 2018.

[3] J. M. Staatz, "A game-theoretic analysis of decision making in farmer cooperatives," *American Journal of Agricultural Economics*, 1983.

[4] N. Ding, R. Erfani, H. Mokhtar, and T. Erfani, "Agent Based Modelling for Water Resource Allocation in the Transboundary Nile River," *Water*, vol. 8, p. 139, Apr. 2016.

[5] M. Zulfiqar and A. Chishti, "Development of Supply and Demand Functions of Pakistan's Wheat Crop," *THE LAHORE JOURNAL OF ECONOMICS*, vol. 15, Jan. 2010.