WHY WOULD YOU DO THAT?

BCS-IITK: Semester Project (2021)

Ayushi Chaudhary, Siddhant Singh, Subiksha Shree S

MENTOR: Shivanshu Tyagi

Github: https://github.com/ayucd/why-would-you-do-that

Project timeline

WEEK 1	Read about reciprocal altruism and the TFT strategy; learnt about the hows-and-whys of the ant-colony-optimisation algorithm.
WEEK 2	Read about evolutionary game theory, learnt about the basics of deep learning and how to work with Pytorch.
WEEK 3	Designed a simulation where agents were trying to survive with a fixed cooperative strategy in an environment.
WEEK 4	Encoded this simulation on Python and drew certain inferences and conclusions [Part A].
	Mid-project evaluation
WEEK 5	In-depth reading on deep learning and learnt how to work on neural networks in Python.
	Break of ~2 weeks
WEEK 6	Read papers on RL, deep RL and multi-agent RL.
WEEK 7	Involved a QL-tweak to our original Python implementation [Part B].
	End-project evaluation

Background

Rational decision-making in a multi-agent environment where kin-based altruism does not exist is governed by inferences drawn from the game theory, the social theory as well as conflict theories. Thus, we can say that if we understand these theories and strategies well, the decisions made by the agents can be predicted. Through this project, we are trying to understand the various rational decisions that can be made by an agent in a multi-agent environment by playing with the various modelling parameters. To do this, we set up an environment on Python with multiple agents suspended within it. We have completed this project in two parts, keeping the same environment but changing the decision-making parameters, and each part provides us with thoughtful observations.

The Environment

We have an environment where multiple agents of various sizes are suspended within. **One iteration in our simulation refers to a single pair of day-time followed by night-time.** The survival of these agents depends upon the food that they can get every day, while evolution is a secondary priority- they can only reproduce if they have a sufficient amount of food, and even this happens based on a fixed probability.

During the day-time, an agent's task is to collect as much food as it could. At night, the agents with excess food *can choose* (depending on their sharing strategy) to share their food with the needy, food-less agents. Only the agents with the larger quantity of food are given the ability to reproduce. If the needy food-less agents are unable to get food even at night, they die. It should be noted that the agents can only ask for food from other agents when they have no food for themselves.

When it comes to sharing their excess food, **our agents can adopt one of the four strategies**, depending on the model chosen-

1. Always cooperative (AC) *Agents following this strategy are assigned the maximum probability to share food and get food in return.*
2. Tit-for-tat (TFT) - *Agents decide whether they will share their food or not based on the history of the agent-at-mercy (needy agent).*
3. Alternatively cooperate (ALT) - *Sharing alternates between cooperativeness and competitiveness for every iteration of gathering food.*
4. Always defective (AD) - *Agents following this sharing strategy are assigned the least probability to share the food they have acquired.*

Python implementation of the environment

Our aim from the Python implementation of this multi-agent environment is to find out which strategy (or strategies) will yield the highest population by the end of our iterations (one iteration is nothing but a pair of consecutive day and night: hence it is an indicator of the time passed since the simulation began). Another thing to observe is to decide which agent size will be having the upper hand in the evolutionary process by the end of our iterations in both models. The implementation has a very probabilistic approach:

- The agents are initialized with a few attributes like *size, id and strategy*. The "agent *id"* is unique for every individual as it reflects the total population that has existed in the environment so far.
- The environment is modelled as an (n x n) matrix. Each cell in the matrix can host either one unit of food or one agent.
- Our simulation will have a day and a night in one iteration. In one round of day and night i.e. one iteration, agents get spawned at random places in an empty grid. Similarly, grids are randomly populated with food on a daily basis.
- The agent that is close to the food gets an advantage over grabbing the food.
- No limit is set over the quantity of food that an individual could acquire. This is done deliberately to observe the sharing strategies that the agents will adopt at the end of the day.
- In cases of conflict, we refer to the priority of directions, which we accomplished by using the breadth-first search (BFS) algorithm. The priority order of the directions is:

```
dirn =[(0,1),(1,0),(0,-1),(-1,0),(1,1),(1,-1),(-1,1),(-1,-1)]
```

- The agents can possess any number of *"food-values"* starting from 0. At the end of the day, if the agent has more than 1 food-value, it gets to choose if it prefers to share it with other needy agents or keep just it for itself.
- Each agent is checked for food at night and three *special cases* can arise:
    I. **Case 1: 0 food-value:** If it has no food it is very much likely to get eliminated. It will not get eliminated if it manages to acquire food from a cooperating/sharing agent (see the next case).
    II. **Case 2: Greater than 1 food-value:** If the agent has more than 1 unit of food, it can decide whether it will share its food or not based on its own strategy; and its decision will also be based upon the past strategies of the agent-at-mercy (needy agent).
    III. **Case 3: Greater than 2 food-values:** In this case, the agent is provided with the ability to reproduce. It can decide whether it will reproduce or not based on a fixed probability provided by us for the model.

PART A (without any Q-learning by the agent)

What sharing strategy thrives the best in our environment

In this part of the project, all agents are following a fixed sharing policy throughout their lifespan. However, to understand the evolution of the agents better, we decided to split our simulation into two different models:

1. **Model 1**: Each agent is allowed to use any one of the four sharing strategies for its entire lifetime. There will be a low (but fixed) chance of reproduction for agents.
2. **Model 2**: The TFT strategy cannot be used by the agents in this model. Thus, they can use only one of the three strategies for their entire lifespan. There will be a (comparatively) higher (and fixed) chance of reproduction for the agents in this simulation. Food quantity is less than that available for Model 1.

Another notable decision-making parameter added in this part of the project is **the size factor**. For this, we are assuming that the agents are of two sizes: Size 1 and Size 2, such that Size 2> Size 1. In both of these models, the agents of Size 2 are going to be selectively altruistic (more willing to share their food) towards needy agents of Size 1. Mathematically, this implies that **the probability of an agent of Size 2 to share its food increases when it meets an agent of Size 1 than when it meets an agent of Size 2**.

Python implementation of Part A

The two models have different, but fixed probabilities for reproduction in Part A. This means that every agent in both models has a fixed chance to reproduce. Following code determines the probability of sharing by an agent having more than 1 unit of food:

```
# code wrapped to fit within the page
prob = (agent.size/(atmercy.size+agent.size))*agent.strat[0]*
(atmercy.strat[-1] if atmercy.strat != [] else 1)
```

After implementing the above mentioned on Python and reiterating the simulation on the 1st model 44 times while a bit lesser (40 times) on the 2nd model, we find ourselves being able to analyze the best strategies for both the models after the iterations. We are also able to figure out which size will be more favoured for survival in the long run for both models (*the number of iterations passed simultaneously represents the time that has passed in this particular environment.)*

Observations

Model 1

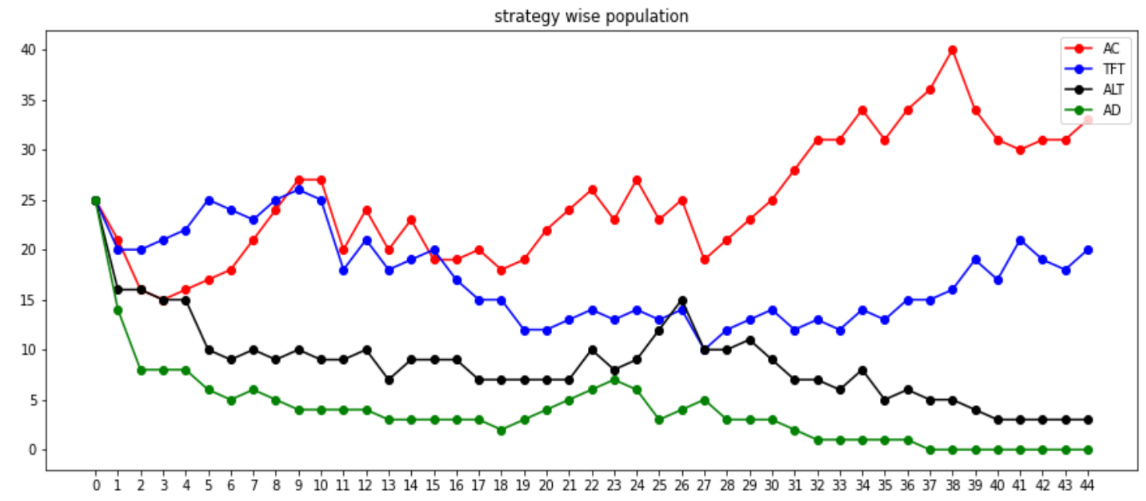

Figure 1: Strategy-wise population development for Model 1, repeated over 44 iterations

We can see that agents following the AC and the TFT strategy are dominant as the environment evolves. The AD and the ALT strategy gradually decline in population.

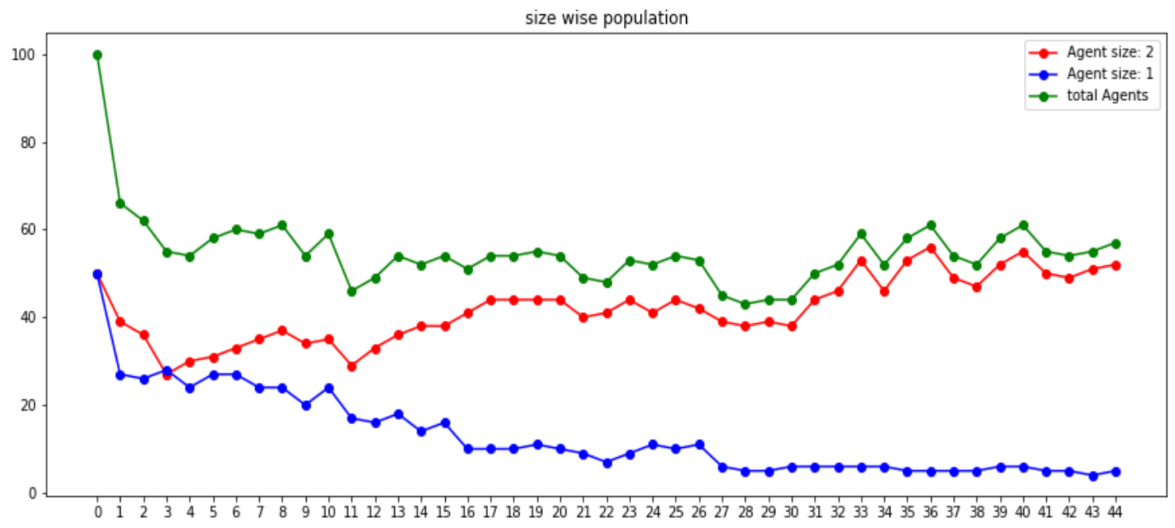

Figure 2: Size-wise population development for Model 1, repeated over 44 iterations

We observe that the environment favoured agents of Size 2 (larger, more altruistic towards Size 1) as the environment evolved with this model.

Model 2

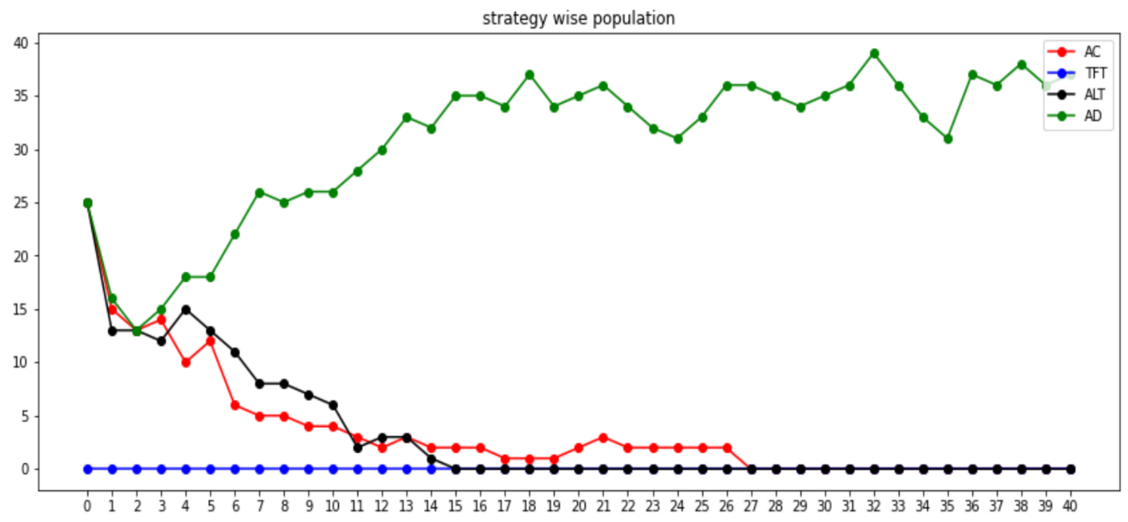

Figure 3: Strategy-wise population development for Model 2, repeated over 40 iterations

We are able to observe that the agents following AD strategy tend to thrive as the environment evolves. The agents following the AC and the ALT strategy tend to die down.

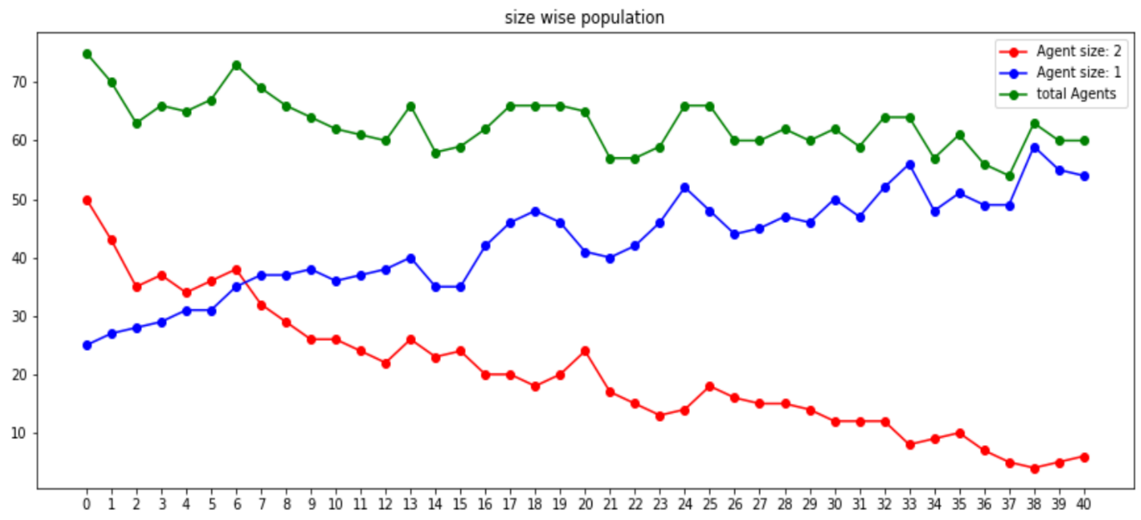

Figure 4: Size-wise population development for Model 2, repeated over 40 iterations

We can see that the agents of Size 1 tend to thrive in this model while the agents of Size 2 (the larger, more altruistic towards Size 1) tend to die down.

PART B (agent performs Q-learning)

Will the agent thrive if left to choose an optimal strategy in the environment?

In this part of our project, we introduced some "special" agents (s-agents) into our original environment, such that they do not follow a fixed strategy, but "learn" which strategy would be the best for their survival. Of course, the environment still contains the primitive agents that are following fixed strategies from the basket of strategies, throughout their lifespan.

When with excess food, the s-agents learn an optimal strategy to survive their way through the environment, by estimating which immediate action is available to it at night during the time of sharing in order to achieve evolution. Thus, Part B is different from Part A in the following ways:

1. Some special agents are allowed to *choose* their strategies.
2. There exists no size based distinction in the agents. All agents are assumed to be of Size 2.

Everything else about the agents and the environment in this part is the same as in Part A (environmental parameters etc.)

Thus, our s-agent can meet 5 types of needy agents at night: the agents following AC strategy, agents following the TFT strategy, agents following the ALT strategy, agents following the AD strategy as well as s-agents following the "intelligent" strategy.

Python implementation of Part B

In this implementation, we are trying to teach our "special" agents how to deal with different strategies. Our special agent learns this via Q-learning.

For analysing the environment, we train our s-agents (obtain the optimal strategy) and then test it. A Q-table is initialised with (states, actions). The Q-table is global in our case, so as to ensure a larger collection of data to refer to. Here,
states= *the strategy of the needy agent that our s-agent is interacting with.*
actions= *strategy chosen by the s-agent to deal with the needy agent.*
After every iteration, our s-agents update the global Q-table.

The s-agents get a **positive reward** if either the population of other agents lowers down as compared to when compared to the previous iteration or its own population increases and vice-versa: thereby **making the environment more competitive**.

We should also take into account that **the current action of the special agents does not lead to the next state**: there is no relation between the action taken by the special agent and the next state because the next state is only obtained in the next iteration on the Q-table and that is random (since our Q-table is global). But we have tried to overcome this by randomising the probability of the next state according to the population of the herd:

```
prob = [0.0,0.0,0.0,0.0,0.0]\
for key in pop_of_strat.keys():
   prob[strat_to_num[key]] = float(pop_of_strat[key][-1])/len
return random.choices(strat_names,prob)[0]
```

So we assume that the next state is correct.

Training the s-agents: Larger number of s-agents are employed to obtain an optimal policy with accuracy. The code is iterated n=44 times and these n iterations are further repeated 5 times to generate a larger amount of data.

Testing the s-agents: A smaller number of s-agents needed. The code is iterated n=44 times.

While testing, our optimal strategy may perform a bit worse but its performance should be largely satisfactory.

Observations

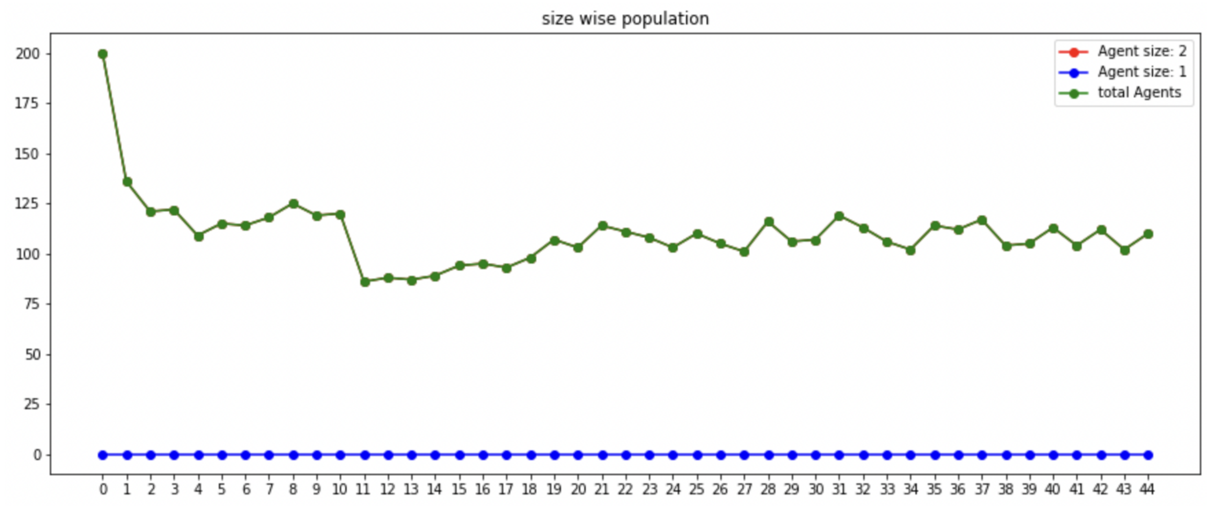

Figure 5: Size-wise population development of our environment, repeated over 44 iterations (testing)

This was expected since Size 1 agents do not exist in the environment anymore.

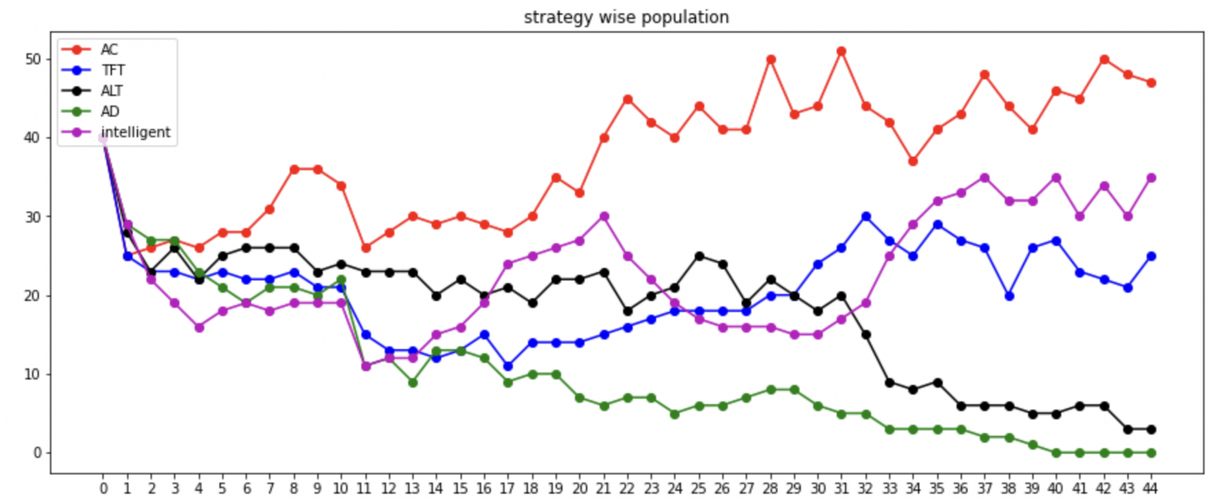

Figure 6: Strategy-wise population development of our environment, repeated over 44 iterations (testing)

We see that the AC strategy thrives amongst the overall population, followed by the "intelligent" strategy and then by TFT. The ALT and AD strategies see a decline in their population as the environment evolves.

Conclusions

Part A

1. **Lack of a punishing mechanism gives rise to cheaters and decline of cooperators within the environment:** TFT strategy in the 1st model in this part assumed a good punishing mechanism for the cheaters, namely the AD agents and sometimes the ALT agents. This conclusion goes hand-in-hand with the conditions that define "reciprocal altruism", which was simulated via the modified (probabilistic) TFT strategy in Model 1. We know that for reciprocal altruism to flourish, it is necessary to have a proper "punishment" mechanism in place so that the detection of "cheaters" in the environment can be facilitated.
2. **Factors that may not be directly relevant to the act of cooperation (here, size), can affect the evolutionary pattern of the environment:** Regardless of the strategies used by the agents for cooperation, there may be other factors that can give them undue advantage over agents which lack them. In our case, it was the size of the agents, which got hard-coded inside our code that determined the probability with which an agent was likely to share its extra food. Thus, in a general multi-agent environment, seemingly irrelevant factors may prove to be disadvantageous for the agents that lack them.

Part B

1. **The ALT strategy becomes vestigial to the special agents:** The ALT strategy that the special agent "can" take, implies no computational meaning when we are actually iterating the model. This is because the agent is free to change its strategy and hence its decision to cooperate or not cooperate in the next iterations, and the possibility to actually "alternate" its decision never really arrives.
2. **The special agent learns a complex sharing-behaviour that is a mix of mostly AC and TFT.** Thus, we can say that our special agent has learnt to survive in an environment via mostly cooperating. The agent learns how to handle each agent within the environment with remarkable accuracy. While it learns to play TFT against AD agents and AC against TFT agents, it also learns that it should cooperate with other special agents.

References:

1. *C. Wickramage and D. N. Ranasinghe*, "Modelling altruistic and selfish behavioural properties of Ant Colony Optimisation," 2014 14th International Conference on Advances in ICT for Emerging Regions (ICTer), 2014, pp. 85-90, doi: 10.1109/ICTER.2014.7083884.
2. *Stephens, C.* (1996). Modelling Reciprocal Altruism. The British Journal for the Philosophy of Science, 47(4), 533-551. Retrieved June 18, 2021, from http://www.jstor.org/stable/687923
3. *Axelrod, R.* (1984). The Evolution of Cooperation. New York: Basic Books.
4. Li, Yuxi. (2018). Deep Reinforcement Learning. Retrieved July 18, 2021, from https://arxiv.org/abs/1810.06339v1