Universal Basic Income Simulation Using RL

Productivity Vs Equality

A PROJECT REPORT

Submitted by :

Laishram Pongthangamba (B19011)
Akash Karnatak (B19147)
Abhishek Mishra (B19231)

For the partial fulfillment of Course

CS672 : Advanced Topics in Deep Learning

School of Computing and Electrical Engineering,

IIT Mandi,

HP - 175005, India

1 . Introduction :

1.1 Aim :

With the recent advancements in the field of AI, there is an ongoing debate of AIs replacing humans in most of the jobs that exist today. With this situation, it will become important for the government to support unemployed people by providing them with a universal basic income(UBI) .

The project aimed at simulating an economy to study the effect of the introduction of AI agents in a world.Our target is to find how agents will react with the introduction of AI agents and to find the optimal value of UBI income that the government should provide to sustain all the people in the economy.

1.2 Motivation :

AIs replacing humans on a large scale may trigger a dangerous situation and may lead to economic collapse as we aren't yet prepared to handle such kinds of situations.
Conducting experiments related to economic matters directly in the real world is often impractical and risky as it can lead to dangerous implications.
Hence, We came up with an idea to simulate the economy, so that we can approximate the real world and find how the situation would be in the upcoming years with AI replacing humans in jobs. The recent discussion around the government providing UBI also inspired us to test its feasibility in this situation.

1.3 Understanding Environment and Work Done:

The basic structure of the environment to simulate the world was provided by salesforce [1]. However we changed the environment a lot to meet our requirements. In the environment we have human_agents, ai_agent and a governor interacting and maximizing their own reward. At any point a typical environment looks something like follows:

Fig : Environment snapshot at arbitrary timestep

Here the light green and white blocks are places where woods and stones are spawned. The circle with a coloured star in it describes the position of various AI agents and various blocks with the same color are houses built by respective AIs. The purple block represents water and AI agents can't go through it. Another thing which also restricts AI movement is houses.

In the below sections we provide an overview of the environment that we used for making this project.

1.3.1 Entities and Components :

- **Labour :** Internal state of an agent. Defines the amount of work the agent does or the cost of performing a task from the agent's point of view.
- **SourceBlock :** Responsible for producing woods and stones. This has been implemented in "WoodTreeSpawn" class and is responsible for producing woods and trees in the environment.
- **House :** Entity built by agents to increase their coins. To an agent building a house costs 1 wood and 1 stone and 1 unit of labor. Depending upon the agent's skill level, it gets a reward in the form of a coin.
- **Water :** Blocking entity to restrict agent's movement. It is represented by a solid block where agent's can't step onto.
- **Coin :** It is representative of money in the world. This we get from trading a resource (low skill) or building a house (high skill).
- **Inventory :** Each agent has its own inventory where he stores all its earned woods, stones and coins.
- **Escrow :** As soon as agent's put its resource for bidding / auction , its resource is moved from inventory to its escrow section . This has been done so as to prevent the agent from using up the resource that he has already declared for the auction.

*****************Code snippet of environment components: ********************

```python
class Resource:
    name = None
    color = None
    collectible = None


    def __init__(self):
        assert self.name is not None
        assert self.color is not None
        assert self.collectible is not None


resource_registry = Registry(Resource)


@resource_registry.add
class Wood(Resource):


    name = "Wood"
    color = np.array([107, 143, 113]) / 255.0
    collectible = True


@resource_registry.add
class Stone(Resource):


    name = "Stone"
    color = np.array([241, 233, 219]) / 255.0
    collectible = True


@resource_registry.add
class Coin(Resource):


    name = "Coin"
    color = np.array([229, 211, 82]) / 255.0
    collectible = False
```

```python
class Landmark:
    name = None
    color = None
    ownable = None
    solid = True
    def __init__(self):
        assert self.name is not None
        assert self.color is not None
        assert self.ownable is not None

        self.blocking = self.solid and not self.ownable

        self.private = self.solid and self.ownable

        self.public = not self.solid and not self.ownable


landmark_registry = Registry(Landmark)


for resource_name in resource_registry.entries:
    resource = resource_registry.get(resource_name)
    if not resource.collectible:
        continue

    @landmark_registry.add
    class SourceBlock(Landmark):


        name = "{}SourceBlock".format(resource.name)
        color = np.array(resource.color)
        ownable = False
        solid = False


@landmark_registry.add
class House(Landmark):

    name = "House"
    color = np.array([220, 20, 220]) / 255.0
    ownable = True
    solid = True
```

1.3.2 Action Spaces for agents :

The action space for agents is discrete. At any step an agent can take any action out of 50 available options based on the policy. The 50 action spaces for agents are as listed :

- It can choose to do nothing and retains his previous position. (1 action.)
- It can move in the grid. In doing this he can choose to move into any of the 4 directions i.e up, down, left, right . In doing so, if the position at which the agent was earlier had a wood/stone, he simply collects that and labor for collecting the resource is deducted from his coins and resource is further added to its inventory. (= 4 actions.)
- He can choose to build a house using his resources (1 wood, 1 stone and 1 labor) . (= 1 action)
- There are 44 options available for trade. He can choose 1 of wood/stone to trade for, then can choose one trading option out of buy or sell. He can then choose 1 value out of 11 possible values (0 -10) as a value at which he wants to trade the resource for. (= 2*2*11 or 44 actions.)

The action space for the governor is however to decide the UBI and tax rate at the beginning of each time step in an episode, and later at the end collecting the coins from agents according to the declared rates and redistributing it among all the agents.

*****************Code snippet of Action spaces OF MOBILE AGENT *****************

```python
class BaseAgent:

    name = ""

    def __init__(self, idx=None, multi_action_mode=None):
        assert self.name

        if idx is None:
            idx = 0

        if multi_action_mode is None:
            multi_action_mode = False

        if isinstance(idx, str):
            self._idx = idx
        else:
            self._idx = int(idx)

        self.multi_action_mode = bool(multi_action_mode)
        self.single_action_map = (
            {}
        )
```

```python
@agent_registry.add
class BasicMobileAgent(BaseAgent):


    name = "BasicMobileAgent"
```

```python
@agent_registry.add
class BasicPlanner(BaseAgent):

    name = "BasicPlanner"

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        del self.state["loc"]
        self._idx = "p"

    @property
    def loc(self):
        """
        Planner agents do not occupy any location.
        """
        raise AttributeError("BasicPlanner agents do not occupy a location.")
```

1.3.2 Rewards for agents:

The utility (or reward) describes the overall happiness of an agent and this is the metric that each agent tries to maximize during training. The reward is a function of Labor and Coin which belongs to the player. For all agents other than governor, the reward can be calculated as :

Reward = utility form coins - disutility from labor

Utility from labor decreases linearly. So the more labor an agent does, more unhappy he becomes. The utility from coins increases in a concave fashion. The coin utility increases rapidly upto a certain point, but after that point the marginal increase is very low as compared to the effort. It follows isoelastic function.

1.3.2.1 Reward(utility) for Human_agents:

As we can see through the above graph, the utility of the agent doesn't increase continuously with the labor hour. He definitely gets more income with the increased labor, but overall happiness decreases which is the same case as the real world. Those dots in the plot represent the optimal point that each skilled agent tries to reach.

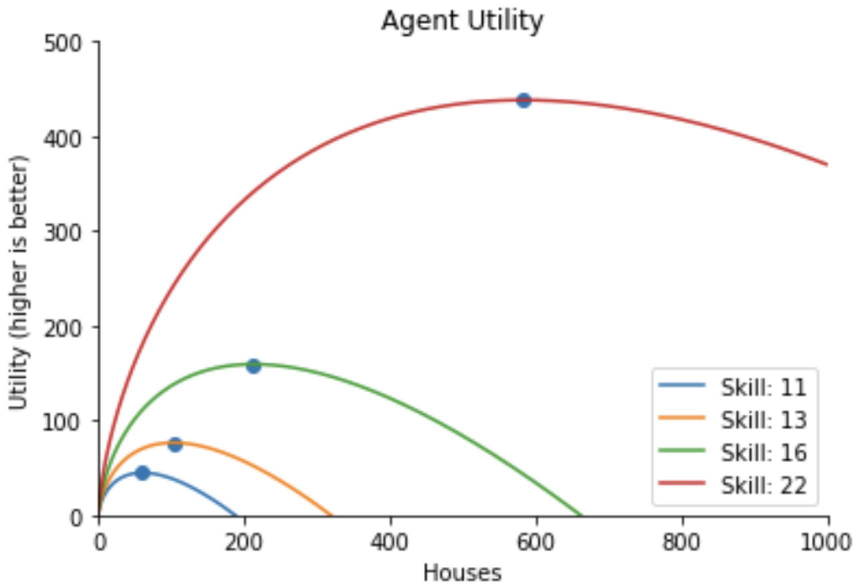


Fig : Reward Curve for human_agents

1.3.2.2 Utility for AI_agent :

An AI can work for long hours without feeling tired or getting sick. So, an AI's reward keeps on increasing with the increased labor as compared to its human counterpart. They have been modeled as human agents with very high skill level so that their reward optimal point lies very high and right side of normal human_agent.[ADD PLOT]

1.3.2.3 Utility for governor :

The prevalence of equality and less disparity among various agents is the reward for the governor. The governor tries to maximize the product of productivity and equality. Productivity has been defined as the total coins earned by all the agents while equality has been defined as the inverse of the difference between coins possessed by the highest earning agent and lowest earning agent.

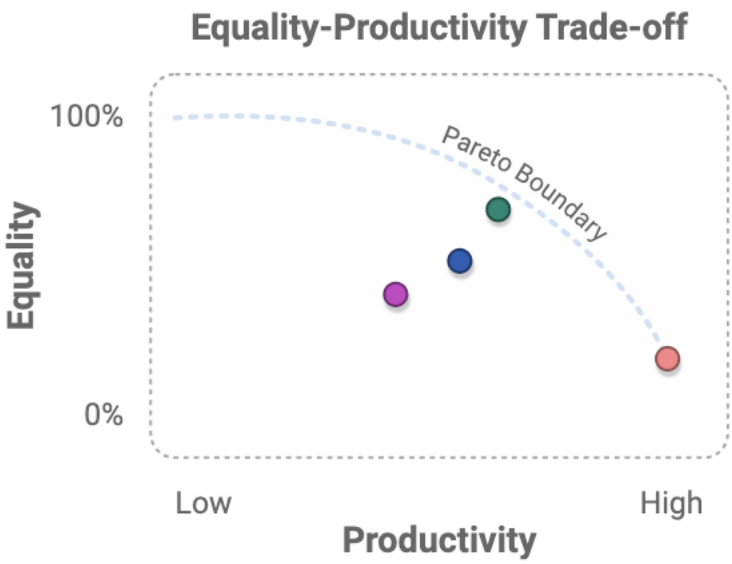


Fig : Equality Vs Productivity tradeoff

This curve describes the utility and aim of the governor i.e is to find the optimal point on the equality vs productivity graph by defining the taxing rates and setting an UBI rate.

*****************Code snippets of UTILITY and REWARD********************

```python
def isoelastic_coin_minus_labor(
    coin_endowment, total_labor, isoelastic_eta, labor_coefficient
):

    assert np.all(coin_endowment >= 0)
    assert 0 <= isoelastic_eta <= 1.0

    # Utility from coin endowment
    if isoelastic_eta == 1.0:
        util_c = np.log(np.max(1, coin_endowment))
    else:
        util_c = (coin_endowment ** (1 - isoelastic_eta) - 1) / (1 - isoelastic_eta)

    # disutility from labor
    util_l = total_labor * labor_coefficient

    # Net utility
    util = util_c - util_l

    return util
```

```python
def coin_eq_times_productivity(coin_endowments, equality_weight):

    n_agents = len(coin_endowments)
    prod = social_metrics.get_productivity(coin_endowments) / n_agents
    equality = equality_weight * social_metrics.get_equality(coin_endowments) + (
        1 - equality_weight
    )
    return equality * prod
```

```python
def compute_reward(self):


    utility_at_end_of_last_time_step = deepcopy(self.curr_optimization_metric)

    self.curr_optimization_metric = self.get_current_optimization_metrics()

    # reward = curr - prev objectives
    rew = {
        k: float(v - utility_at_end_of_last_time_step[k])
        for k, v in self.curr_optimization_metric.items()
    }

    self.prev_optimization_metric.update(utility_at_end_of_last_time_step)

    avg_agent_rew = np.mean([rew[a.idx] for a in self.world.agents])

    if avg_agent_rew > 0:
        self._auto_warmup_integrator += 1

    return rew
```

1.3.3 Agents :

1.3.3.1 Human_agents :

These agents are representative of the human workforce in an economy whose sole aim is to maximize their own reward(utility).
To simulate the differences that each human possesses in terms of skills and his ability to earn, we assign a skill level to each AI agent which is sampled from a pareto distribution. Based on their skill they can do less paying jobs such as collecting stones and woods and later trading them or high paying jobs such as buying collected woods from low skill labors through trading and later building houses through these materials.

1.3.3.2 AI_agents :

These are similar to human_agents except for their longer capability to do continuous work and having a different utility (reward) function as compared to human_agents (see figure : [utility curve of ai_agent]).

1.3.3.3 Governor :

The Governor is the representative of the government. Its aim is to promote equality while not compromising with the productivity of the agents. The governor looks into each of the episodes to see how each agent is playing and how much income he is making. Governor collects a proportion of money from each agent based on its income and redistributes the wealth among all the agents. For unemployed agents this distributed wealth acts as UBI and helps them to survive in an economy while being unemployed.
The task of the governor isn't as easy as it seems. The rate of taxation should be carefully decided because overtaxing may discourage working agents from earning more while taxing less will increase inequality among the society and may prove fatal for the unemployed_agents.

******************Code snippet of RL PPO IMPLEMENTATION******************

```
def build_trainer(run_configuration):
    trainer_config = run_configuration.get("trainer")
    env_config = {
        "env_config_dict": run_configuration.get("env"),
        "num_envs_per_worker": trainer_config.get("num_envs_per_worker"),
    }
    if trainer_config["seed"] is None:
        try:
            start_seed = int(run_configuration["metadata"]["launch_time"])
        except KeyError:
            start_seed = int(time.time())
    else:
        start_seed = int(trainer_config["seed"])

    final_seed = int(start_seed % (2 ** 16)) * 1000
    logger.info("seed (final): %s", final_seed)
    dummy_env = RLlibEnvWrapper(env_config)
    agent_policy_tuple = (
        None,
        dummy_env.observation_space,
        dummy_env.action_space,
        run_configuration.get("agent_policy"),
    )
    planner_policy_tuple = (
        None,
        dummy_env.observation_space_pl,
        dummy_env.action_space_pl,
        run_configuration.get("planner_policy"),
    )
    policies = {"a": agent_policy_tuple, "p": planner_policy_tuple}
    def policy_mapping_fun(i):
        if str(i).isdigit() or i == "a":
            return "a"
        return "p"
    if run_configuration["general"]["train_planner"]:
        policies_to_train = ["a", "p"]
    else:
        policies_to_train = ["a"]
    trainer_config.update(
        {
            "env_config": env_config,
            "seed": final_seed,
            "multiagent": {
                "policies": policies,
                "policies_to_train": policies_to_train,
                "policy_mapping_fn": policy_mapping_fun,
            },
            "metrics_smoothing_episodes": trainer_config.get("num_workers")
            * trainer_config.get("num_envs_per_worker"),
        }
    )
    def logger_creator(config):
        return NoopLogger({}, "/tmp")
    ppo_trainer = PPOTrainer(
        env=RLlibEnvWrapper, config=trainer_config, logger_creator=logger_creator
    )
    return ppo_trainer
```

2 . Technical Details :

2.1 Algorithm Used :

Proximal Policy Optimization (PPO) was used to learn the optimal policy for each agent. PPO belongs to the class of policy optimization methods and is considered state-of-the-art in Reinforcement Learning. This aims to find an optimal policy by directly optimizing the policy function itself, rather than estimating the value function.
The reason for choosing PPO over other algorithms like DQN and DDQN was that PPO has been found to perform better in most scenarios as compared to DQN. It also produces more stable training to prevent fluctuation during training. One other reason for preferring PPO over others was its capability to strike balance between exploration and exploitation by utilizing a clipped surrogate objective. This allows for conservative policy updates that

prevent the policy from deviating too far from the current policy, thus providing a better exploration-exploitation trade-off.

Following is summary of the algorithm and its training procedure taken from the original paper :

Algorithm 4 PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k)$$

  by taking K steps of minibatch SGD (via Adam)
  if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} = 2\beta_k$
  else if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \leq \delta/1.5$ **then**
    $\beta_{k+1} = \beta_k/2$
  end if
end for

PPO, penalty variant. The objective function embeds a penalty on the KL-divergence. The corresponding weight β_k is dynamically updated based on the measured divergence relative to the target divergence [source: Schulman et al. 2017]

Fig : PPO Algorithm

2.1 Connecting Pieces:

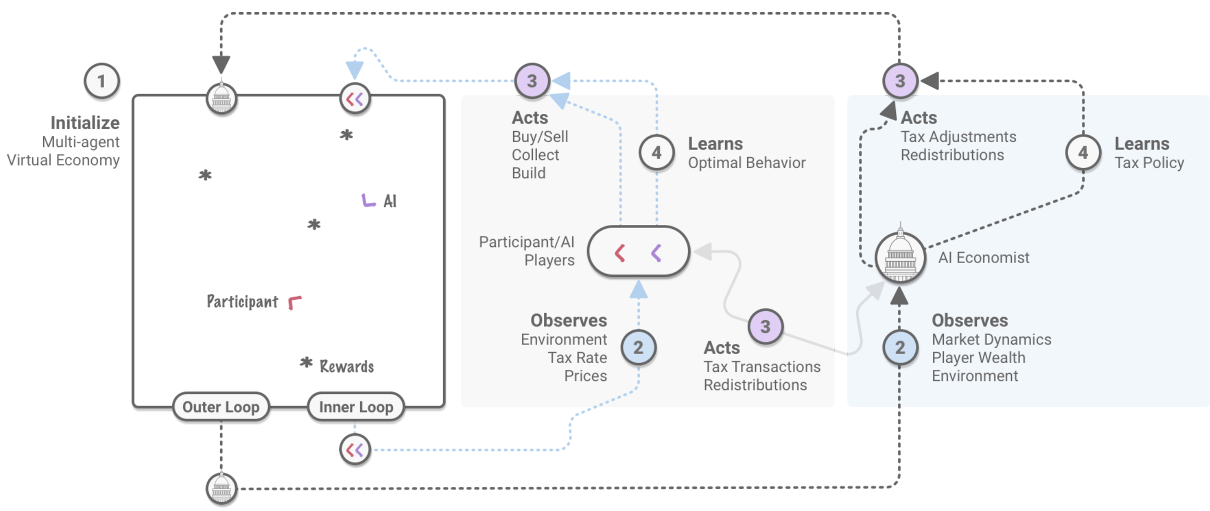


Fig : Connecting Pieces [Src : 1]

The agents are spawned . [NEED MOST TECHNICS].

3. Results and Observation :

To simulate the project we are using 3 human_agents, 1 AI agent, and 1 governor (if applicable).

The skills levels are:

- Human agents - The skill levels of human agents are sampled from pareto distribution.
- AI agent - The skill level of an AI agent is approximately 8 times the highest skill level among humans.

3.1 Free Market:

First we simulate our RL agent in a free market where there is no taxes and ubi . Here the RL agents explore the surroundings and try to maximize their own utilities.

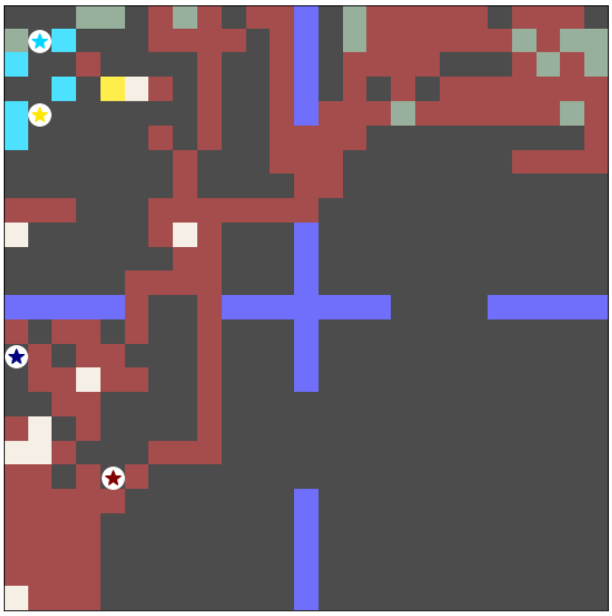


Fig. 3.1 Free market environment.

3.1.1 Snapshots:

The following are the snapshots taken while simulating at different times.
* Brown color agent is the AI agent and others (Blue, Yellow, Light Blue) are human agents.

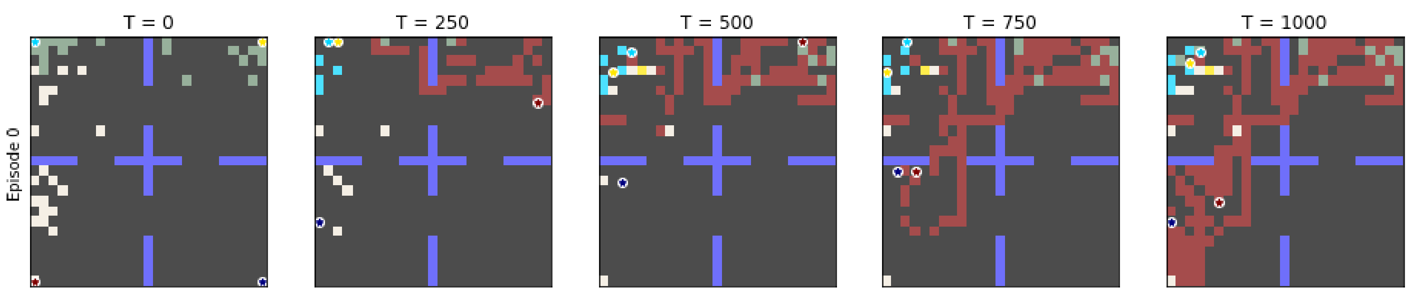


Fig: 3.2 - Snapshots of environment at different timestamps.

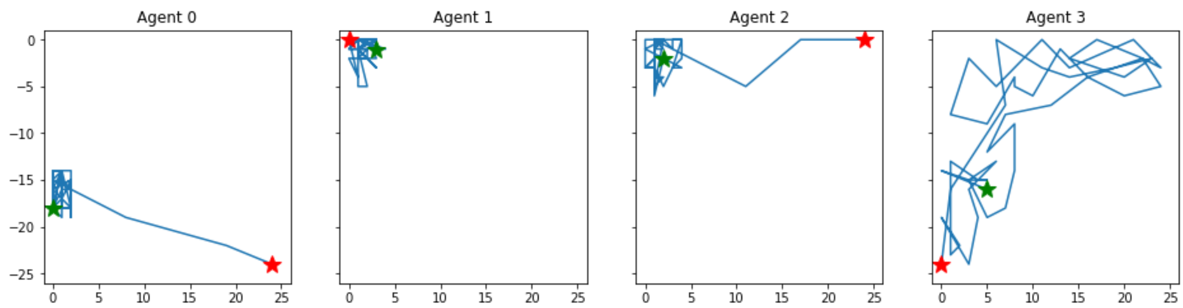


Fig : Movement of different agent during the episode.

Observations:

- At the starting, agents spawn at the corners then they start exploring the environment.
- The AI agent which has the highest skill makes a large number of houses and starts surrounding the resources to protect them from other agents. At T = 1000, we can see it even block movement of other agents by building houses around them.
  Here we see in the free market , in order to maximize its own happiness (utility) it tries to capture all the resources and block other agent movements.
- Other agents try to build but their houses are negligible as compared to AI agents.

3.1.2 Trading and collecting of resources for each agent:

	Agent 0	Agent 1	Agent 2	Agent 3
Cost (Wood)	4.83 (n= 6)	4.60 (n= 10)	6.00 (n= 2)	5.23 (n=139)
Cost (Stone)	4.00 (n= 6)	3.87 (n= 23)	3.72 (n= 18)	4.33 (n=129)
Income (Wood)	5.17 (n= 6)	5.25 (n= 75)	5.15 (n= 65)	4.91 (n= 11)
Income (Stone)	4.15 (n=106)	4.21 (n= 33)	4.31 (n= 36)	4.00 (n= 1)
Income (Build)	~~~~~~~~	16.47 (n= 5)	11.33 (n= 1)	40.00 (n=152)

Here , n is the total number of entities(wood, stone or house) that an agent buys , sells or builds.

We see in the free market -

- Agent 3 (AI): It buys almost all of the stone and wood and uses them in building houses.
- Agent 0(Human): It specializes in gathering mostly for stone as he collects 106 stones and sells them.
- Agent 1(Human) and Agent 2(Human): They build houses also but earn more income from gathering and trading.

3.1.3 Resources usage and accumulation over time for each agent:

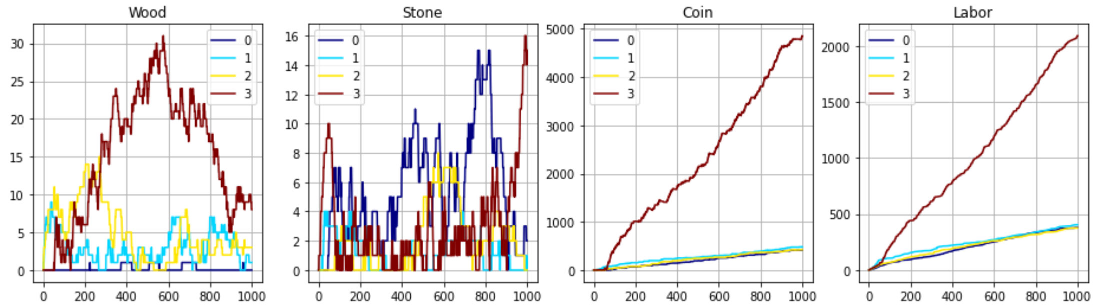


Fig : Trading and resource collection by each agent

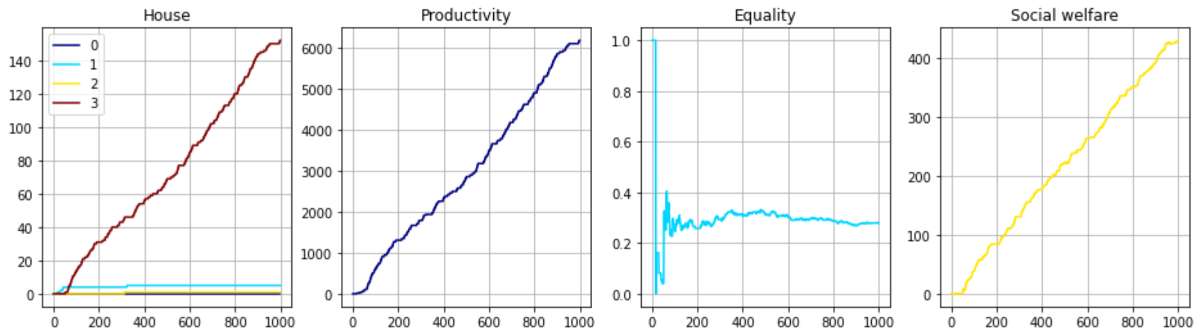


Fig : Houses , Productivity , Equality and Social Welfare.

Observations:

- At the start of the episode, Agent 3 (AI) starts buying more wood but as soon as it buys stones are being used in building houses. Number of houses, coins and labor increase with time.
- Most coins of the other agents come from trading.
- Overall productivity increases very large up to 6000 but equality remains at 0.25 which is very low. Most of the wealth is held by Agent 3(AI).
- Social welfare seems to increase due to very high productivity. But Social welfare of 400 only is very low as compared in case of taxes which is 1000.

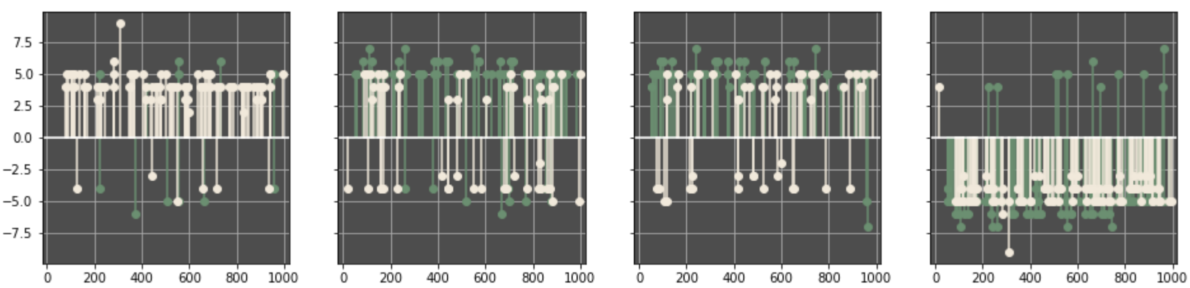


Fig : Trading stats (White bars represent stones and green depicts woods. The bars above the axis denotes selling while below the axis are buying instances)

3.2 Market being Regulated by Governor:

We noticed in the free market the AI agent was dominating all the fields and human_agents were hardly earning anything as compared to the AI agent.
In the second scenario we allowed the governor to regulate the market by imposing taxes on the agents and redistributing the collected coins among all the agents. This distributed wealth acts as UBI for the agents and promotes equality within the society.

3.2.1 Training Snapshots:

After all the training was done, following was the state of the environment with agents :

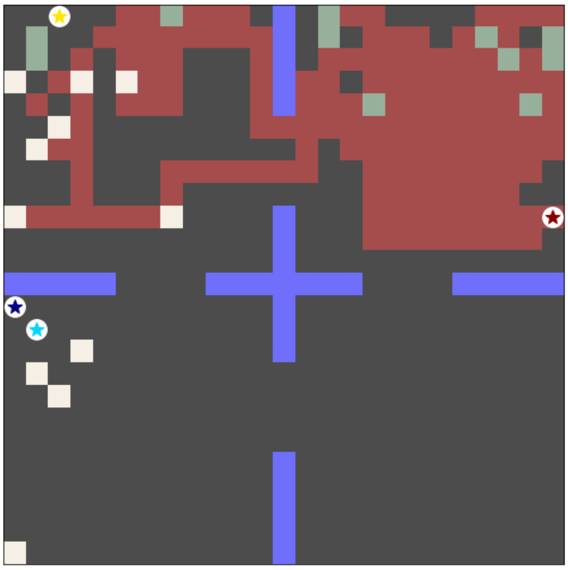


Fig : Final Environment after training in the presence of governor

Fig : Training snapshots at regular interval

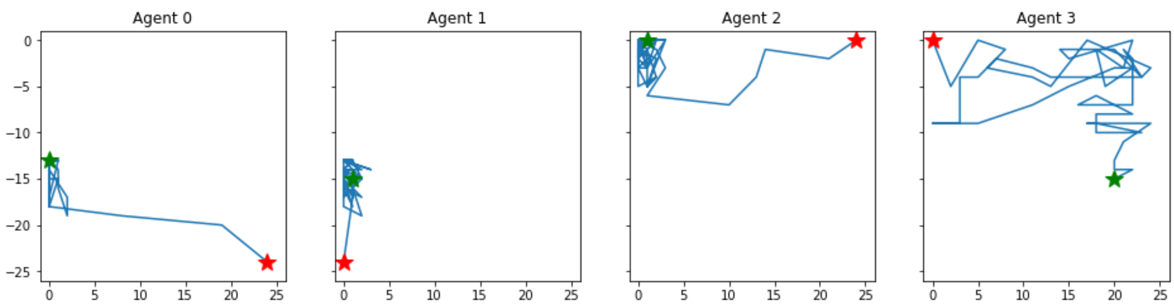


Fig : Movement plot of agents

Following were the noted observations :

- We can observe through the training snapshot and final result that the AI agent is trying to get hold of all the places where resources are available. It also tries to block other agents from moving and entering its territory by making houses on their way.
- But we also observed that since AI agent had less money as compared to free market case, the extensiveness of the blocking wasn't very large.

3.2.2 Trading and collection of resources by each agent:

	Agent 0	Agent 1	Agent 2	Agent 3
Cost (Wood)	4.07 (n= 30)	4.16 (n= 25)	4.27 (n= 15)	4.40 (n=131)
Cost (Stone)	5.33 (n= 18)	5.21 (n= 14)	5.11 (n= 9)	5.63 (n=156)
Income (Wood)	4.13 (n= 30)	4.36 (n= 25)	4.33 (n=132)	4.43 (n= 14)
Income (Stone)	5.67 (n= 58)	5.56 (n= 84)	5.41 (n= 54)	5.00 (n= 1)
Income (Build)	~~~~~~~~	~~~~~~~~	~~~~~~~~	40.00 (n=162)

Fig : Trading and resource collection by each agent

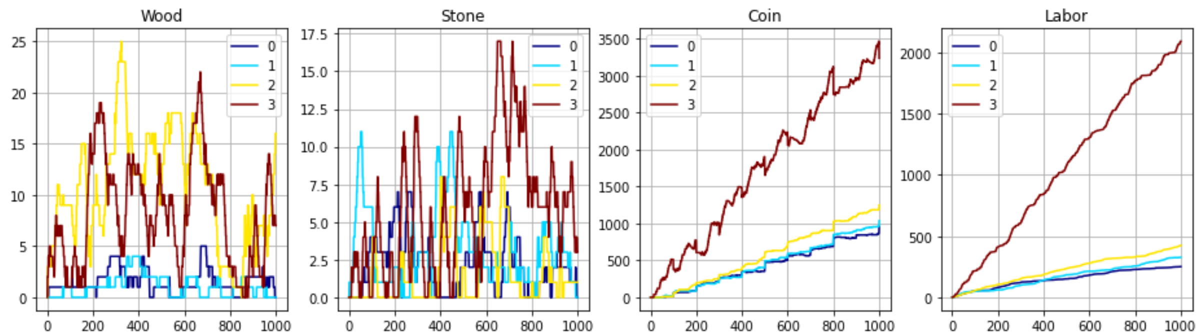


Fig : Visualization Resources change over time

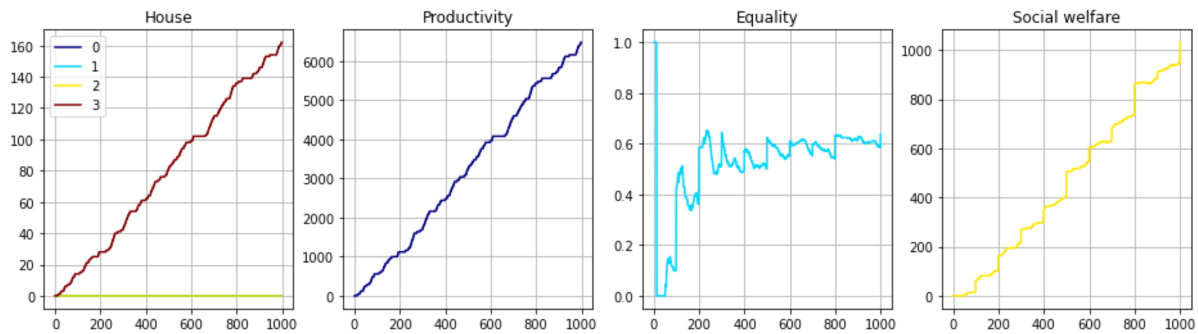


Fig : Visualization various metrics over the ime

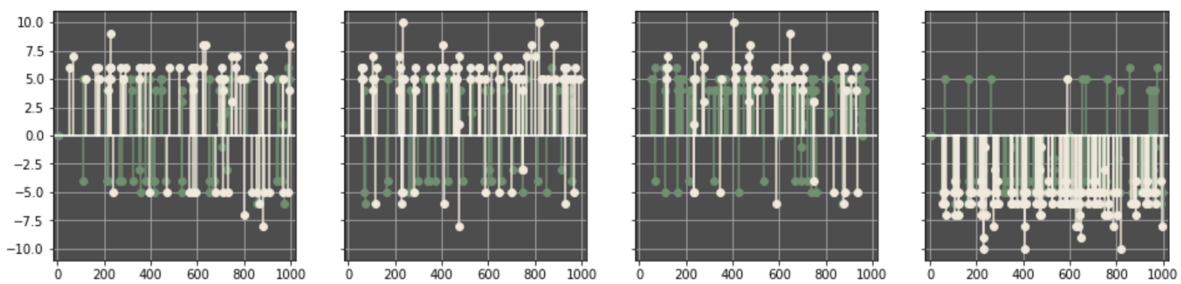


Fig : Trading stats (White bars represent stones and green depicts woods. The bars above the axis denotes selling while below the axis are buying instances)

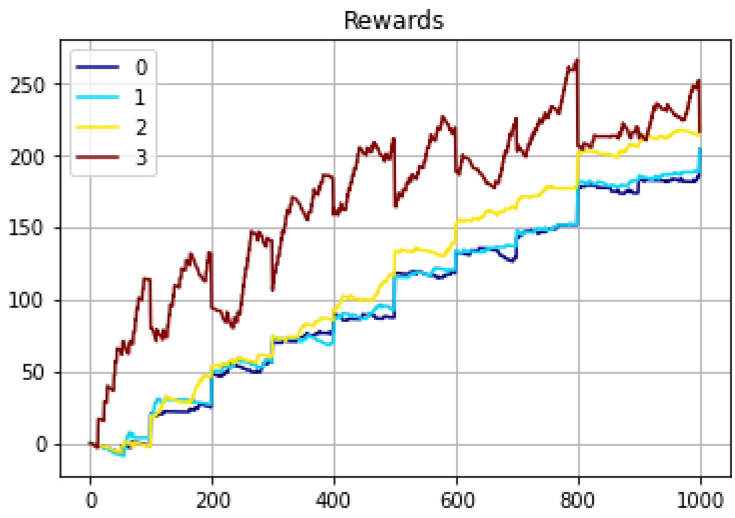


Fig : Visualization reward for each agent

Following observations were made :

- From the cost rows of 1st figure, we noticed that A_ agent bid more money as compared to human_agents who were more conservative of spending money and buying resources. AI agent bid higher price for the resource to eliminate competition and try to buy as many resources as possible. He later utilizes the collected resources to build houses which gives higher returns as compared to money spent on buying the resources.. Human_agents on the other hand try to buy the resource at the minimum price and hence are able to build a lower number of houses.
- From the income rows of the figure1 we found that AI specialized in high skill and high paying jobs of making houses. In Fact all houses were built by AI_agent only. Human agents on the other hand tries to specialize in lower skills jobs like collecting stones and woods and later selling them to AI_agent. The AI agent itself indulges very less in collecting resources by himself, and prefers to buy it from lower skilled human_agents.
- We also observed that agent1 specialized in collecting and selling stones, while agent2 specialized in collecting and selling woods while ai_agent specialized in building houses. However agent0 didn't specialize in anything and that's why he couldn't generate a good income as compared to other human_agents and became almost unemployed.
  Agent0 buyed wood collected by agent2 at lower cost and later sold it at higher price and similarity did agent1.
- From the 2nd graph of coins, we observed that income of the ai_agent sees a dip after regular steps, and regular increase in coins of lower skilled agents. This happens because of collecting taxes from the agent and redistributing the UBI to the lower agents.
- We also observed that ai_agent was capable of doing higher labor as compared to human_agent. What surprised us was the behavior of agent0 . He initially tried to do some labor of reselling the woods or collecting some stones and woods. But he didn't get specialized , so he was earning less. But once it learned of the UBI, it stopped laboring as visible in the final part of the labor plot of 2nd figure.
- From the 3rd figure and comparing with the corresponding counterpart in the free market , we observed that with the implementation of UBI and taxes increased the equality to 60% as compared to 25% of the free market. We also observed that UBI

also helped to achieve the total social welfare of 1000 points as compared to 400 in case of the free market.

3.2.3 Governor (Planner):

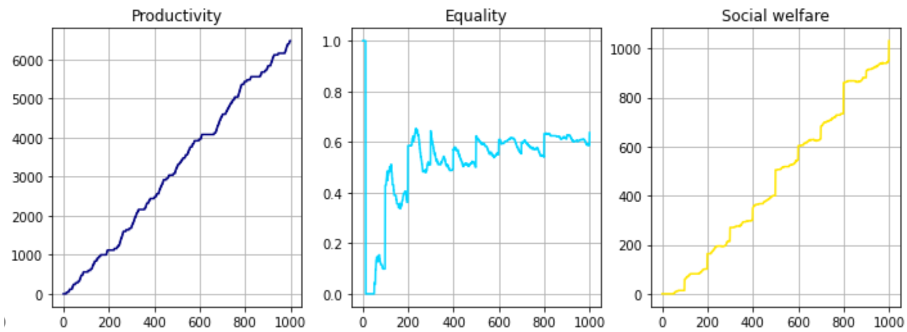


Fig :Productivity , Equality and Social Welfare.

Period 1:

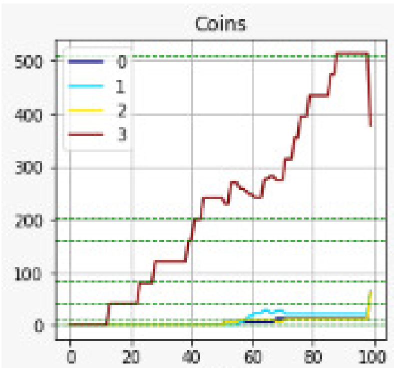


Fig : Coins possessed by different agents in 1st period.

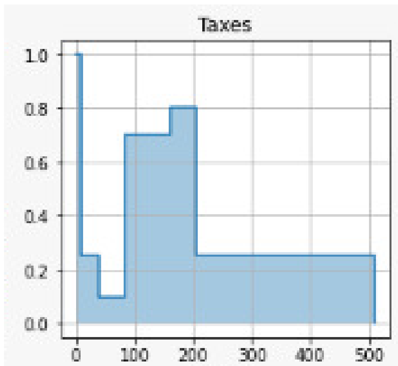


Fig : Taxes policy set by planner

Period 10:

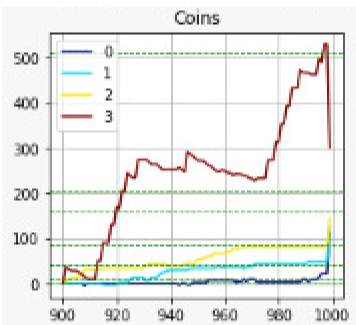


Fig : Coins possessed by different agents in 10th period.

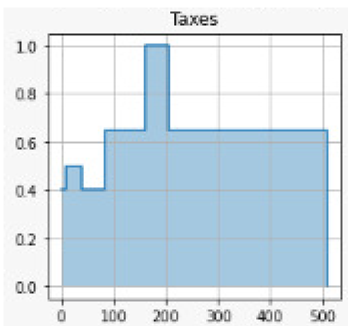


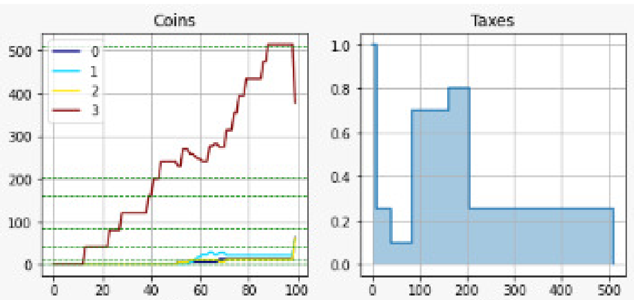
Fig : Taxes policy set by planner

The governor(planner) looks at all the coins collected by all agents and tries to bring quality by imposing taxes.

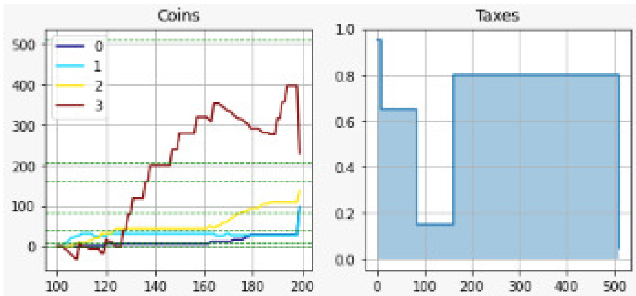
Observations:

- In the starting periods, the number of coins of all the agents are similar, so the planner focuses on increasing productivity, so the planner takes less tax in higher income.
- At the end of the period , the planner realizes that there is a huge difference in income so from next period it tries to increase the equality by increasing tax on the high income group and increasing the UBI.
- But there are so many fluctuations in the tax policy over a period.
- Overall the planner brings equality at about 0.6 and tries to maintain it . As well as trying to increase productivity in order to increase social welfare.
- In the last period, it tries to make two main sections by applying a large tax in the middle.

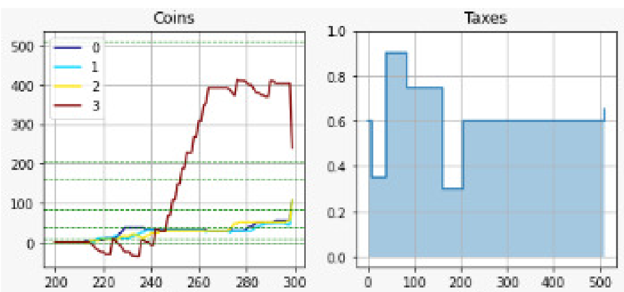
Period Wise coins and taxes:



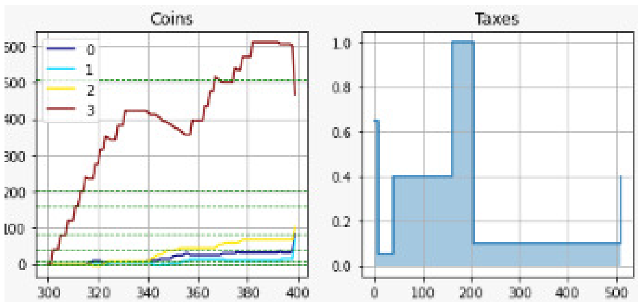
Period1



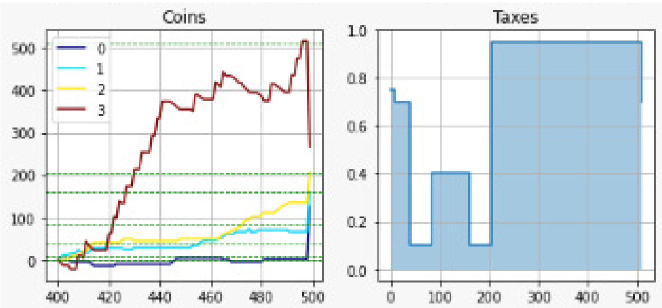
Period 2



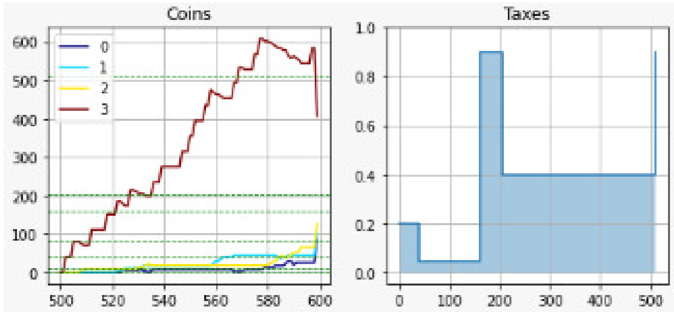
Period 3



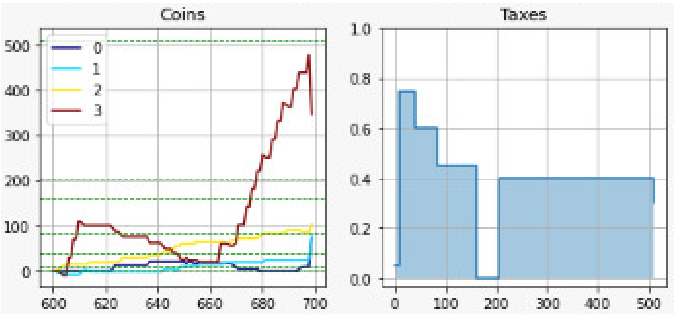
Period 4



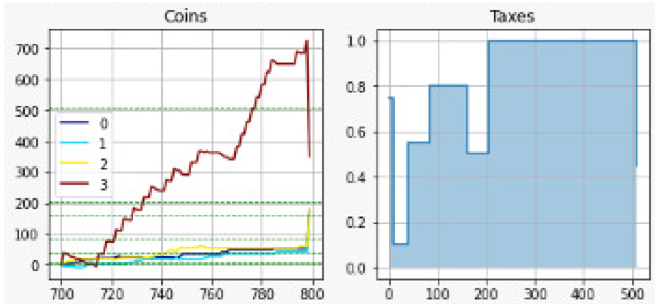
Period 5



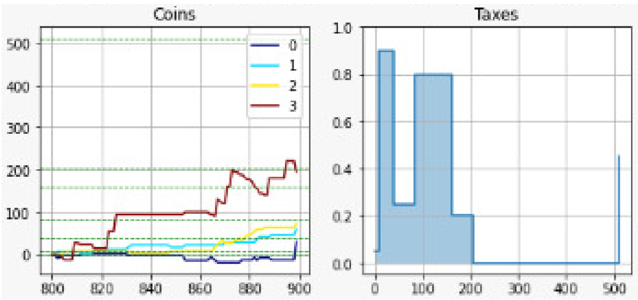
Period 6



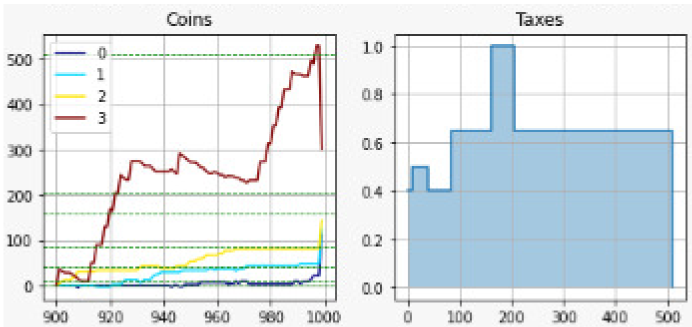
Period 7



Period 8



Period 9



Period 10

4. Conclusion:

In the simulated economy with AI agents replacing humans, the AI agent with the highest skill concentrated resources and power, while human agents focused on lower-skilled jobs. Implementing a Universal Basic Income (UBI) and taxes increased equality and social welfare. The planner adjusted tax policies to address income disparities and aimed for an equality level of around 0.6. However, stable tax policies were challenging to maintain. Overall, the UBI influenced agent behavior and showed potential for mitigating inequality and promoting social welfare in an AI-dominated economy.

5. References:

1. https://blog.salesforceairesearch.com/the-ai-economist/
2. https://en.wikipedia.org/wiki/Isoelastic_utility
3. https://en.wikipedia.org/wiki/Gini_coefficient
4. https://docs.ray.io/en/latest/rllib/rllib-env.html
5. https://spinningup.openai.com/en/latest/algorithms/ppo.html