Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Nome corso di Laurea

A STUDY ON TAXATION POLICY ON HETEROGENEOUS NON-ANALITICAL BEHAVING AGENTS

Relatore:
Chiar.mo Prof.
NOME RELATORE

Presentata da:
NOME LAUREANDO

Sessione
Anno Accademico

Questa è la Dedica*:*
ognuno può scrivere quello che vuole,
anche nulla . . .

Introduction

hi this is my intro

Contents

List of Figures

List of Tables

Chapter 1

What is Foundation

In this chapter I am going to introduce the package provided by Salesforce called Foundation. This package offers the possibility to create simulations with multiple agents that interacts in a 2D world. First a presentation of the the Gather and Trade simulation setup will be purposed, followed by a description of the agents involved, their action set, observation and scope.

descrivi il capitolo quando lo hai finito

1.1 Gather and trade

The simulation that will be used throght the dissertation is called gather and trade. This simulation takes place in a 2D map that represent the world where the agents lives and interact. The shape of the world is a 25x25 grid where are disposed various kinds of entities. Within this world 4 agents are free to move around, gather resources, trade them and use them to build houses. These agents are different for their skill level, allowing them to have higher/lower rewards for their actions. A fifth agent, called policy maker, is asked to tax and redistribute the income of the 4 previous agents, based on informations about their endowments, but not their skill.

1.1.1 World and entities

As said before the map is a 25x25 grid that where are present some entities. Some of these are visible in the world, others are just present in the agents endowment.

- Water
- House
- Source Block (Wood and Stone)
- Wood and Stone
- Coins
- Labor

Water is a simple world block that has no use other than avoiding agents to pass through. We can see from Figure 1.1 that the water is used to divide the world in 4 macro areas, each one with different resources. A House is a block that is not present at the beginning of the simulation, but it can be built by agents in any empty block, agents can't walk through houses either.

Sorce blocks are two entities that spawns stochastically resources, namely wood and stone, as we can see from Figure 1.1 in the four areas divided by water we have a zone with both wood and stone source block, two other areas with just one kind of source block and the last one that is empty.

Coins is the measurement of the value produced in the world. Coins are generated when a house is built, the agent that builds the house is rewarded with a certain amount of coins that variates with the skill level.

Labor is a measurement of the total effort exherted by an agent, this is generated every time an agent takes an actions and generates disutility.

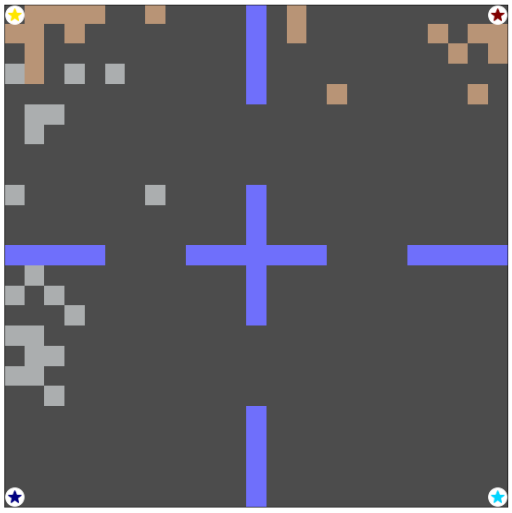

Figure 1.1: Rendering of the world at inital conditions: *this is a rendering of the world map at the first timestep of the simulation, blue block are water, brown blocks are wood source, grey block are Stone source. The four corners are the starting positions for the 4 agents.*

1.1.2 Game dynamics

A problem that has a continuous flow of agent-environment interactons can be formalized by a Finite Markov Decision problem [1]. In particular this simulation is a partial-observable multi-agent Markov Games (MGs). The problem is defined by the tuple $(S, A, R, \text{simbolo}, \gamma, o, \mathcal{T})$ where S is the state space and A is the action space.

The simulation is composed by a series of episodes, each of length H timesteps. At every point in time $t \in [0, H]$ the episode is characterized by a state s_t, representing the current world environment, every agent performs an action $a_{i,t}$ given the current partial observation $o_{i,t}$ of the world state, and recives a reward r_{it}. Afterwards the environment transits to a new state s_{t+1} according to the transition distribution $\mathcal{T}(s_{t+1}|s_t, \boldsymbol{a}_t)$. This chain of interactions state-action-reward/state carries on until the end of an episode.

$$s_0 \rightarrow_{\boldsymbol{o}_0} \boldsymbol{a}_0 \rightarrow r_1, s_1 \rightarrow_{\boldsymbol{o}_1} \boldsymbol{a}_1 \rightarrow \ldots \rightarrow_{\boldsymbol{o}_{H-1}} \boldsymbol{a}_{H-1} \rightarrow r_H, s_H$$

Here $\boldsymbol{a}$ and $\boldsymbol{o}$ are the vectorized observations and actions for all the five agents. Given the particular structure of the simulation every single agent will recive an observation at every timestep (different for everyone, more on that later), but only at the 4 basic agents will be asked to perform an action, the policy maker will act only upon a certain condition met. In this case the episode last for 1000 timestep and the policy maker is asked to act (tax the other agents) every other 100 steps. The existence of multiple episodes is necessary for the 4 agents and the policy maker to define their own optimal policy $\pi_i(o_{i,t}, h_{i,t-1}; \theta_i^*)$, this optimization process will be the focus of chapter RL.

1.1.3 Agents

From what above we know that the four basic agents are endowed with labor, coins, wood and stone. They live in the world map, can act within it and their objective is to maximize their γ-discounted utility function. Now I will describe in more in detail the agents starting from the inforamtions that they receive at each timestep, then talking about the actions that they are allowed to take and finally about their objective.

Observation space: Given that this simulation is a partial-observable multi-agent Markov Game, the observation that agent i receive at time t is not complete but partial, this inforamtions can be summarized in the following way:

- $o_{i,t}^{\text{world state}}$: world map situation surrounding the agent, this is limited to the 11x11 grid around the agent i.
- $o_{i,t}^{\text{market state}}$: full information about the market state for wood, stone and available trades.
- $o_{i,t}^{\text{agent}}$: public resouces and coin endowments (this information is also available to the policy maker) and private labor performed and skill level.

- $o_{i,t}^{\text{tax}}$: tax structure
- $o_{i,t}^{\text{other}}$: other inforamtions (ex. action mask)

the full observation space can be seen in Table A.2

Action space: The agent can take one action per timestep and can choose this action from the 50 listed below:

- **Movement**: 4 actions for the basic movements N, S, E, W
- **Gather**: 1 action for gahtering
- **Trade**: 44 actions for trading resouces
- **Idle**: 1 action that do notthing

The movements actions along with gather do not need much of an explaination, these are simple actions that costs a quantity of 0.21 labor units each time pursued. The building action require the agent to consume (destroy) one unit of wood and one unit of Stone, as a consequence he gains 2.1 units of labor and an amount of coin that depends on his skill level. The most complicate set of actions are the one that rules trading. Each one of them is a combination of the 11 price levels [0,1,...,10] that the agent is willing to (pay/request) for each side (bid/ask) and for each resource (wood/stone). A trading open action remains open for 50 tunrs, if in this time span it is matched by the corresponding counter action at the same price (a bid for an ask and vice versa) then the trade takes place and each agent gets a 0.05 units of labor.

The action mask, present in the observation space, is a touple of binary values of length 50 that "masks" inconclusive actions, such as moving north while at the north border of the map, or building a house without the required wood and stone. This is used in the learning process to avoid wasting time in exploring meaningless actions.

Agent objective Agents in the simulation earn coins when building houses or trading goods,

The utility for the four agents is an isoelastic utility:

$$u_i(x_{i,t}, l_{i,t}) = crra(x^c_{i,t}) - \vartheta_k l_{i,t}\,, \quad crra(z) = \frac{z^{1-\eta} - 1}{1-\eta}\,,\ \eta > 0 \tag{1.1.1}$$

Where $l_{i,t}$ is the cumulative labor associated with the actions taken up to time t, $x^c_{i,t}$ is the coin endowment and ϑ is a function utilized for labor anihilation with the following specification $\vartheta_k = 1 - exp\left(-\frac{episode\ completitions}{energy\ warmup\ constant\ (k)}\right)$. This variable will play an important role during the two step optimization process purposed in the original paper. In particular during the phase 1 of training the labor cost is annihilated to help agents avoid sub-optimal behaviours. And η determies the degree of nonlinearity of the utility. This utility function is assumed to be the same for all the agents.

The maximization problem is solved for a rational behaving agent by optimizing the total discounted utility over time,

$$\forall i\ :\ \max_{\pi_i} \mathbb{E}_{a_i \sim \pi_i, \boldsymbol{a}_{-i} \sim \boldsymbol{\pi}_{-i}, s' \sim \mathcal{T}} \left[\sum_{t=1}^{H} \gamma^t r_{i,t} + u_i(x_{i,0} l_{i,0}) \right] \tag{1.1.2}$$

with $r_{i,t} = u_i(x_{i,t}, l_{i,t}) - u_i(x_{i,t-1}, l_{i,t-1})$ being the istantaneous reward of agent i at time t. Equation 1.1.2 illustrates a multi-agent optimization problem in which actors optimize their behavior at the same time, since the utility of each agent is dependent on the behavior of other agents. Another agent, for example, may deny an agent access to resources, limiting how many houses the agent can create in the future and hence its utility. While computing equilibria for complicated environments like this is still out of reach, we will see in RL chapter how RL may be utilized to produce meaningful, emergent behaviors.

1.1.4 Policy maker

The policy maker, or social planner, differs deeply from the previous agents. Being the focus of the research question it's structure and behavior changes a lot in every single simulaion.

Observation space: The observation space of the social planner depends on the simulation, for most of the simulations no observation is needed.

se riesci a fare RL anche su di lui descrivi l'obs space di quella simulazione

Action space: the action space is quite similar amongs all the simulations, the social planner has to decide how much to tax the individuals according to their total income. If the policy maker is

non è facile finchè non ho deciso le simulazioni da fare ... magari lo rimado a dopo

Chapter 2

Reinforcement Learning

Reinforcement learning is learning what to do to maximize a numerical reward. A discovery process where the learner is not told what action to take, but instead must discover which action yields the most reward by trying them. In the most interesting cases actions might affect not only the immediate reward but also future situations and all the subsequent rewards. These two characteristics – Trial-and-error search and delayed reward – are the two most important distinguishing features of reinforcement learning.

Reinforcement learning differs from *supervised learning* since training is not based on an external dataset of labeled examples, where each situation (observation) is labelled with the correct action to perform (often identify a category). RL, although one might erroneously think the opposite, is also different from *unsupervised learning*. The main objective for unsupervised learning is to find hidden structures in an unlabeled dataset, whereas RL's main objective is to maximize a reward signal.

2.1 Finite Markov Decision process (MDPs)

Finite Markov decision processes are a class of problems that formalize subsequent decision making, where not only the influence of the action is exerted

on immediate reward but also on those in the future. MDP's are suited for RL since these are models that frame the process of learning through repeated interaction between an agent (decision maker), and an environment (ruled by fixed state transition function).

More specifically an agent is asked to take an action $a_t \in \mathcal{A}$ at every time step $t = 0, 1, \dots$. To do so the agent is provided with an observation of the current environment's state $s_t \in \mathcal{S}$ and a reward $r_t \in R$ from the previously performed action. Afterwards the environment update it's state following a transition distribution $\mathcal{T}(s_{t+1}|a_t, s_t)$ and a numerical reward $r_{t+1} \in \mathcal{R} \subset \mathbb{R}$. This process is reproduced every subsequent timestep, this concatenation of interaction is a MDP.

Objective and Rewads: The main objective of RL is to maximize the total number of rewars it receive. This reward r_t is passed from the environment to the agent at every timestep as a consequence of his actions. In the case of the Gather and Trade the reward is $r_{i,t} = u_i(x_{i,t}, l_{i,t}) - u_i(x_{i,t-1}, l_{i,t-1})$.

Since the agents wants to maximise the total upcoming reward we can write this value simply as the sum of all the future rewards.

$$U_t \doteq r_{t+1} + r_{t+2} + \dots + r_H,$$

Where H is the total lenght of the episode, and at the timestep $t = H$ the episode ends. This state is called the terminal state and is a particular state because regardless of the final condition of the agent it reset the environment to the initial condition and restart completely the episode. Another specification for the total reward can implement the concept of discounting, which is more appropriate in the case of economical simulations, thus within the experiments the agent has to maximise his future discounted utility:

figure 3.1 from [1]

$$U_t \doteq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{H-t-1} r_H, \tag{2.1.1}$$

the discount rate $\gamma \in [0, 1]$ determines the value that the agent assign to the future reward, a reward recived at k timesteps in the future is only valued γ^{k-1} times what it would be valued today. When the value of γ approaches 0 the agent is more "myopic" and puts most of his interest in immediate rewards, while if it approaches 1 the interest is more projected in the future due to the stronger impact of furure rewards.

Value functions: this one might be a subparagraph One of the most important elements of RL is the value function. The estimation of this function is one of the crucial points of RL, in fact it tries to quantify the expected return of the rewards it expects to receive. Furthermore the expected reward depends on the action that the agent decides to take, thus the value function are defined in terms of policies, which are acting behaviors. Formally a policy is

If the agent is following policy π at time t, then $\pi(a|s)$ is the probability that the agent take the action $a_t = a$ given the state $s_t = s$. The aim of RL is to change the policy based on experience across episodes to find an optimal behavior.

Now we can write a value funciton for a state s unther the policy π. This function is the expected return when the inital state is s and the policy π is followed from thereon.

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[U_t | s_t = s\right] = \mathbb{E}_\pi \left[\sum_{k=0}^{H} \gamma^k r_{t+k+1} \middle| s_t = s \right], \quad \text{for all } s \in \mathcal{S} \tag{2.1.2}$$

$v_\pi(s)$ is called the state-value function for policy π Following from this equation is possible to define the value of thaking an action a in the state s following the policy π:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[U_t | s_t = s, a_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{H} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right], \tag{2.1.3}$$

and $q_\pi(s,a)$ is called the action-value function for policy π. The important concept here is that the value functions in 2.1.2 and 2.1.3 can be estimated from experience. There are multiple ways to evaluate these functions, we can divide these ways in two main groups, tabulat solution methods and approximate solution methids. For the former we have Montecarlo methods qui metti delle referenze cosi fai una bibliografia grossa ;), Dynamic programming, Temporal-difference learing, n-step bootsrap and others. While for the latter we have -... . . .. and policy gradient methods.

For the purpose of this thesis we are going to foucs only on policy gradient methods and a particular set of optimization policy called proximal policy optimization.

2.2 Approximate solution Methods

The approximate solution methods are a set of strategies tought for those problems, such as ours, where the set of possible states is enormous. It is very likely that every state encountered in a simulation will never have been encountered before. Thus to make sensate decisions there is the need to be able to generalize from previous state that are, to some extent, similar. This is accomplished

2.2.1 Policy gradient Methods

Chapter 13 RL: an into

Policy gradient methods are a set of parameterized policy that can select actions without the use of a value function

2.2.2 Proximal Policy Optimization Algorithms

Proximal policy optimization algorithms (PPOs) are a family of policy gradient methods for reinforcement learning, which alternate between sampling data through interaction with the environment and optimizing a "surrogate" objective function using stochastic gradient ascent.

Chapter 3

Optimal taxation

Appendix A

Tabelle e cose

Parameter		Value
Episode Lenght	H	1000
World height		25
World width		25
Resources respawn prob.		0.01
Inital Coin endowment		0
Iso-elastic utility exponent		0.23
Move labor		0.21
Gather labor		0.21
Trade labor		0.05
Build labor		2.1
Base build payout		10
Max skill multiplier		3
Max bid/ask price		10
Max bid/ask order duration		50
Max simultaneous open orders		5
Tax period duration		100
Min bracket rate		0%
Max bracket rate		100%

Table A.1: Your caption.

Variable Name	Dimension	Bounds
world Map	(7, 11, 11)	{0;1}
world-idx_map	(2, 11, 11)	{0,1,...,5}
world-loc-row	(1,)	[0,1]
world-loc-col	(1,)	[0,1]
world-inventory-Coin	(1,)	[0,inf)
world-inventory-Stone	(1,)	
world-inventory-Wood	(1,)	
time	(1, 1)	
Build-build_payment	(1,)	
Build-build_skill	(1,)	
ContinuousDoubleAuction-market_rate-Stone	(1,)	
ContinuousDoubleAuction-price_history-Stone	(11,)	
ContinuousDoubleAuction-available_asks-Stone	(11,)	
ContinuousDoubleAuction-available_bids-Stone	(11,)	
ContinuousDoubleAuction-my_asks-Stone	(11,)	
ContinuousDoubleAuction-my_bids-Stone	(11,)	
ContinuousDoubleAuction-market_rate-Wood	(1,)	
ContinuousDoubleAuction-price_history-Wood	(11,)	
ContinuousDoubleAuction-available_asks-Wood	(11,)	
ContinuousDoubleAuction-available_bids-Wood	(11,)	
ContinuousDoubleAuction-my_asks-Wood	(11,)	
ContinuousDoubleAuction-my_bids-Wood	(11,)	
Gather-bonus_gather_prob	(1,)	
action_mask	(50,)	

Table A.2: Full observation space.

Bibliography

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.