Ubinet, Distributed Optimization and Game Theory 2016-2017 February 1, 2017

Lecture 7

Lecturer: Giovanni Neglia *Scribe: Yassir M'rabet, Thibaud Canale*

NOTE: The content of these notes has not been formally reviewed by the lecturer. It is recommended that they are read critically.

1 Introduction

This final lesson is about how to use game theory in engineering. The goal of game theory is to optimize a function that is unknown, in many cases we named it the utility function.
We will see that in order to design a system that we want to reach an optimum configuration, we need to deploy agents that can choose between a set of strategies, each giving a certain payoff. Agents will keep changing their actions in order to increase their utilities and it will make them reach an optimal state which is an equilibrium.

2 Potential Games

We want to design a game that converges by dynamics to an equilibrium. We introduce the notion of Potential Games. A game is said to be a potential game if the incentive of all players to change their strategy can be expressed using a single global function called the potential function. When an agent will change its state to increase its utility, the potential function associated to the system will also increase. And in potential games the existence of pure strategy Nash equilibrium is guaranteed, and multiple Nash equilibrium may exist. We only need to choose the potential function that satisfies:

$$\exists\varphi(\underline{a}) \qquad / \qquad U_i(a_i', \underline{a}_{-i}) - U_i(a_i, \underline{a}_{-i}) = \varphi(a_i', \underline{a}_{-i}) - \varphi(a_i, \underline{a}_{-i}) \qquad \forall a_i', a_i, \underline{a}_{-i} \forall i \tag{1}$$

The non-trivial assumption of the equation is that it should be true for all agents.
We will look at the implications of equation (1) in the following. Let us consider a set of agents a_i that can choose to change their state. a_1 will first change its state:

$$a_1^{(0)}, a_2^{(0)}, a_3^{(0)} ... a_N^{(0)} \qquad \text{to} \qquad a_1^{(1)}, a_2^{(0)}, a_3^{(0)} ... a_N^{(0)}$$

It means that a1 increases its utility while it decided to change its state.

$$U1(a_1^{(1)}, a_2^{(0)}, a_3^{(0)} ... a_N^{(0)}) > U1(a_1^{(0)}, a_2^{(0)}, a_3^{(0)} ... a_N^{(0)})$$

Which means, from equation (1), that a1 increased the potential of the system by changing state.

$$\varphi(a_1^{(1)}, a_2^{(0)}, a_3^{(0)} ... a_N^{(0)}) > \varphi(a_1^{(0)}, a_2^{(0)}, a_3^{(0)} ... a_N^{(0)})$$

Now agent a2 changes its state, we get $a_1^{(1)}, a_2^{(1)}, a_3^{(0)} ... a_N^{(0)}$
It means that its action changed the total utility positively:
$U2(a_1^{(1)}, a_2^{(1)}, a_3^{(0)} ... a_N^{(0)}) > U2(a_1^{(1)}, a_2^{(0)}, a_3^{(0)} ... a_N^{(0)})$
Note that we do not know how the utility of agent 1 changed $U1(a_1^{(1)}, a_2^{(1)}, a_3^{(0)} ... a_N^{(0)})$? $U1(a_1^{(1)}, a_2^{(0)}, a_3^{(0)} ... a_N^{(0)})$
But all what matter at this state is the variation of the potential function, and we know by the definition of a potential game that it increased:
$\varphi(a_1^{(1)}, a_2^{(1)}, a_3^{(0)} ... a_N^{(0)}) > \varphi(a_1^{(1)}, a_2^{(0)}, a_3^{(0)} ... a_N^{(0)})$
A question that we can ask ourselves is do the dynamic stop ?
At this point we know that if the dynamic stop we are in a pure strategy Nash Equilibrium. The

condition for the system to stop is that the space of possible strategies should be finite to say that the dynamics will stop. Namely the function of potential φ should have a finite number of values that it can take.

3 Designing a Game

We know the condition that must be satisfied when choosing the utility function in order to be in the case of potential games, which is choosing a function where if each agent changes its state to increase its utility, it should cause an increase in the potential function associated to the utility function.
A good way to make sure that the condition will be satisfied is to choose the potential function to be the same as the utility one $\varphi = W$ therefore the game can be expressed as:

$$\underset{a_i \in A_i \forall i}{Maximize} \qquad W(a_1, a_2, a_3, ..., a_N) \tag{2}$$

The total utility is then computed by:

$$U_i(a_i', a_{-i}) - U_i(a_i, a_{-i}) = W(a_i', a_{-i}) - W_i(a_i, a_{-i}) \tag{3}$$

To be able to compute values of the utility we need to add that at a given point the utility is null, it comes as a referential for the computation. The computation of a difference between potential only include the final state and the initial one, this is why we need to have an information about the first state.
We put $\quad U_i(a_i^0, a_{-i}) = 0 \quad \forall a_{-i}$

4 An Application

We will consider the problem of an area that needs to be covered by robots that sense a specific range. In this area there are risky places where a fire could start. We therefore need to put the robots in the area and then the robots will keep changing until reaching a configuration where risky places are well covered.

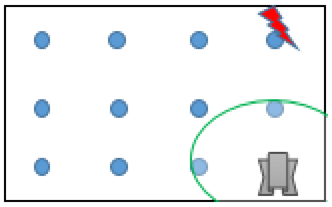

Figure 1: Illustration of the problem

We say that:
A_i is the set of positions.
$P(\underline{s})$ is the probability that the fire occurs at $\underline{s}$.
$D_i(a_i, \underline{s})$ is the probability that robot i at position a_i detects the fire in $\underline{s}$.
Putting these variable, we will define the function that we want to maximize, which is the probability that the robot senses the fire and that a fire occurs:

$$W(\underline{a}) = \sum_{\underline{s}} (1 - \prod_i (1 - D_i(a_i, \underline{s}))) \tag{4}$$

Robots will compute their utility following the equation:

$$U_i(a_i, a_{-i}) = U_i(a^0, a_{-i}) + W(a_i, a_{-i}) - W_i(a^0, a_{-i}) \tag{5}$$

Where: $U_i(a^0, a_{-i}) = 0$
From (4) equation (5) becomes:

$$U_i(a_i, a_{-i}) = \sum_{\underline{s}} P(\underline{s})[1 - \prod_{j \neq i}(1 - D_j(a_j, \underline{s}))(1 - D_i(a_i, \underline{s})) - 1 + \prod_{j \neq i}(1 - D_j(a_j, \underline{s}))(1 - D_i(a^0, \underline{s}))] \quad (6)$$

By looking at this equation we could think that a robot needs to know the position of all the other robots in order to compute its utility, but actually it is not the case, there are places not affected by the sensing of the robots, they will naturally not appear in the equation, we will modify the limits of the sum:

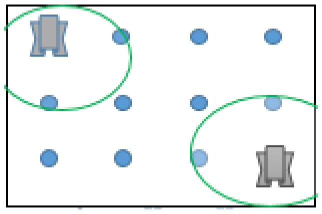

Figure 2: Interaction model between robots

The resultant equation:

$$\begin{aligned} U_i(a_i, a_{-i}) = & \sum_{\underline{s} \in N(a_i)} P(\underline{s})[1 - \prod_{j \neq i}(1 - D_j(a_j, \underline{s}))(1 - D_i(a_i, \underline{s})) - 1 + \prod_{j \neq i}(1 - D_j(a_j, \underline{s}))(1 - D_i(a^0, \underline{s}))] \\ & - \sum_{\underline{s} \in N(a_i)} P(\underline{s})[1 - \prod_{j \neq i}(1 - D_j(a_j, \underline{s}))(1 - D_i(a^0, \underline{s})) - 1 + \prod_{j \neq i}(1 - D_j(a_j, \underline{s}))(1 - D_i(a_i, \underline{s}))] \end{aligned}$$

Another remark is that, as we said, we only look at the sensing region at the initial state and the final one, the equation will therefore not contain all the places where the robot that changes its state is, we will write the equation again but specifying this region by modifying the limit of the product:

$$\begin{aligned} U_i(a_i, a_{-i}) = & \sum_{\underline{s} \in N(a_i)} P(\underline{s})[1 - \prod_{j/\underline{s} \in N(a_j)} (1 - D_j(a_j, \underline{s}))(1 - D_i(a_i, \underline{s})) - 1 + \prod_{j/\underline{s} \in N(a_j)} (1 - D_j(a_j, \underline{s}))(1 - D_i(a^0, \underline{s}))] \\ & - \sum_{\underline{s} \in N(a_i)} P(\underline{s})[1 - \prod_{j/\underline{s} \in N(a_j)} (1 - D_j(a_j, \underline{s}))(1 - D_i(a^0, \underline{s})) - 1 + \prod_{j/\underline{s} \in N(a_j)i} (1 - D_j(a_j, \underline{s}))(1 - D_i(a_i, \underline{s}))] \end{aligned}$$

We rearrange the equation:

$$\begin{aligned} U_i(a_i, a_{-i}) = & \sum_{\underline{s} \in N(a_i)} P(\underline{s})[\prod_{j/\underline{s} \in N(a_j)} (1 - D_j(a_j, \underline{s}))(D_i(a_i, \underline{s}) - D_i(a^0, \underline{s}))] \\ & - \sum_{\underline{s} \in N(a_i)} P(\underline{s})[\prod_{j/\underline{s} \in N(a_j)} (1 - D_j(a_j, \underline{s}))(D_i(a^0, \underline{s}) - D_i(a_i, \underline{s}))] \end{aligned}$$

The result of this equation will give the agent the information on the position it should got to. An output of the computation could be that a place is risky but already well covered by agents, and an agent's increase of utility will be to go to another place not as risky but less covered.
We saw that at first it appears to be a centralized problem where an agent needs to know about the position of all agents to compute its utility, but finally an agent only needs to know about its neighbors, which puts the solution as a distributed one.

5 Full Information Dynamics

In this section, we have now access to the full information of the instance.
"Full Information" means we know **every previous steps** which lead to the current position.

$$a_i(t+1) = F_i(\underline{a}(0), \underline{a}(1), ..., \underline{a}(t), U_i(.)) \tag{7}$$

- a_i is the position for the agent designed by i,
- t is the current step (time),
- U is the utility for the targeted agent, following the function of utility .,
- and F is the function which will compute the best next position, hence the Full Information Dynamics, thanks to knowing all the solutions of all the agents and the function of utility.

6 Oracle-based Dynamic

In this case, we calculate the next step, with this time using an oracle to calculate the utility of every previous position for every other position but ours:

$$a_i(t+1) = F_i(U_i(., \underline{a}_{-i}(0)), U_i(., \underline{a}_{-i}(1)), ..., U_i(., \underline{a}_{-i}(t))) \tag{8}$$

Notes: a_-i means all positions a_j for $j! = i$
In this case, we don't need to know what people are playing, we need to know if I need to play something different and therefore, what will be my payoff if I play such a given position?
BR Dynamic falls in this class: the last step $(a_i(0), a_i(1), ..., a_i(t))$ could be limited to: $a_i(t+1) = F_i(U_i(, a_i(t)))$

7 Payoff based Dynamic

Now we consider all those informations:

$$V_i(a_i, t) = \frac{\sum_{k=1}^{t} U_i(a_i(k)), \underline{a}_{-i}(k) * \mathbb{1}(a_i(k) = a_i)}{t} \tag{9}$$

According to this function, we know **what we will get** when we pick this position, but we only know when we **actually** go on it.
So, we can wonder: **is there other way to learn over time? To finally learn how to go to NE?**

Let's start with full information: Is there a way a learn to play to NE?

This strategy is called **fictitious play**:
"Every player thinks the other players are playing according to a stationary mixed strategy" **This statement is not true!**
See Table 1 below.
Over time, we notice the distribution is 1/3.

We calculate the probability q with the following equation:

$$q_i^j(a_j) = \frac{\sum_{k=1}^{t} \mathbb{1}(a_j(k) = a_j))}{t} \tag{10}$$

Player a_i sees player j is playing its action a_j, where a_j belongs to the set A_j of the following possible actions, R, S, P.

q(2, R) =	0	0	0	1/4	2/5	3/6	$\to$	1/3
q(2, S) =	1	1/2	1/3	1/4	1/5	1/6	$\to$	1/3
q(2, P) =	0	1/2	2/3	2/4	2/5	2/6	$\to$	1/3
1	R	R	S	S	S	P		
2	S	P	P	R	R	R		
q(1, R) =	1	1	2/3	2/4	2/5	2/6	$\to$	1/3
q(1, S) =	0	0	1/3	2/4	3/5	3/6	$\to$	1/3
q(1, P) =	0	0	0	0	0	1/6	$\to$	1/3

Table 1: Example: Rock-Scissor-Paper

	A	B
A	$0,0$	$1,1$
B	$1,1$	$0,0$

Table 2: Normal form of a game

$a_j(n)$ belongs to A_j action j playing at turn n.
a_i belongs to BR (for every j, plays according to q_i^j).
For potential games, $q_i^j(a_j) \to q *^j (a_j)$ $\underline{\ }\, q*^j_{\ j=1..n}$ is a NE.
Good news: if $\underline{q*}^j j = 1..n$ is a pure strategy NE, for t large enough, then $\underline{a}(t)$ is a NE.
Examples:
See Table 2 above.
See Table 3 below.

But the payoff is always zero!

In oracle-based game: an estimate

$$V_i(a_i, t) = \frac{\sum_{k=1}^{t} U_i(a_i(k)), \underline{a}_{-i}(k) * \nVdash(a_i(k) = a_i)}{t} \tag{11}$$

it's called the **Joint Action Fictitious Play**.

8 Joint Action Fictitious Play (JAFP)

Based on the frequential play, you select at every step, you check the following rules:

	1	1/2	2/3	2/4	$\to$	1/2
	0	1/2	1/3	2/4	$\to$	1/2
R	A	B	A	B		
C	A	B	A	B		
	1	1/2	2/3	2/4	$\to$	1/2
	0	1/2	1/3	2/4	$\to$	1/2

Table 3: Example: Rock-Scissor-Paper with JAFP

- if $a_i(t)$ belongs to $argmax_{a_i}(V_i(a_i,t))$ then $a_i(t+1) = a_i(t)$: "you play it again"
- else "you toss a coin":
  - $\rightarrow$ w.p $\frac{1-\epsilon}{(|B_i(t)|)}$, then pick $a_i(t+1)$ in $B_i(t)$
  - $\rightarrow$ w.p ϵ: $a_i(t+1) = a_i(t)$

ϵ is called the **inertia of the player**.

POTENTIAL GAME which is also GENERIC (there are no same action which give the same pay-off).
Joint Action Fictitious Play (JAFP) converges w.p 1 to a NE.

At every time, you have:

- $a_i(t)$ reference action,
- $U_i(t)$ reference utility.

With this information, we can build:

- w.p $1-\epsilon$: $\rightarrow a_i(t+1) = a_i(t)$; $U_i(t+1) = U_i(a_i(t+1), a\underline{-}i(t+1))$
- w.p ϵ: $\rightarrow$ playing a random a_i', which belongs to A_i ; $a_i(t+1) = a_i' =$
  - $(ifU_i(a_i', a_-i(t+1)) > U_i(t))$ then $a_i(t+1) = a_i'$; $U_i(t+1) = U_i(a_i', a_-i(t+1))$
  - (else) then $a_i(t+1) = a_i(t); U_i(t+1) = U_i(t)$

For every $p < 1$, there is an unique $\epsilon > 0$, such as

- for t large enough:
  - $a_i(t)$ is a NE with probability at least p.

if you want to be sure you are playing NE:
you need to pick $p \uparrow 1$; $\epsilon \downarrow 0$

If you explore less, you take more time
All this dynamic guaranty you to reach NE.
The worst case is NE can't be reach.

Price of Anarchy: the worst possible NE can't be worst than twice the optimum.

Another approach is to do Simulated Annealing.
Logistic learning.