A new direction

We noticed that in most of the models that we read (cf. [Drago and Turnbull, 1991], [Chakravarti et al., 2015]), the workplace reward maximizing game that was played is static and there are no dynamics involved. However, in real scenarios, when someone joins a new company, the newcomer does not immediately decide to purely cooperate or compete, similiarly, this is true for his new colleagues too. After some interaction with each other over a time span, everyone will have a rough idea of everybody's work style, ethnics and most importantly the reward derived. This will then allow the agent to decide whether to cooperate or compete with each agent respectively which we shall call it the nature of the agent. Such a model would be a more realistic scenario as it gives each agent time to learn about the environment before determining his nature.

Definition 0.1. The *nature* of an agent in a workplace reward maximizing game with k agents is a vector $\mathbf{x_i} = (x_1^i, x_2^i, \ldots, x_k^i)$ such that $\sum x_i = 1$. It is a discrete probability distribution where x_j^i denotes the probability of cooperating with agent j and x_i^i denotes the probability of not cooperating.

With the definition 0.1 we see that a *completely cooperative* agent i is one with $x_i^i = 0$ and a *completely competitive* agent has $x_i^i = 1$. A game where all the agents are completely cooperative is called a *completely cooperative environment* and if all the agents are completely competitive, it is a *completely competitive environment*.

We would now like to introduce some form of dynamics in which the players learn whether it is to their benefit to cooperate or compete with a given agent. Here we will approach it in a similar fashion of *no-regret dynamics* where we want to minimize regret in single decision-maker playing game against an adversary. In our setting the adversary is the other $k-1$ agents and its ability to extend it to multi-player games and relation to coarse correlated equilibria is how we plan to model our game. (We would then vary the different reward systems to see how it will cause the CCE to change and whether we can design a reward system with the dominant-strategy incentive-compatible (DSIC) for workplace reward maximizing games.)

The model

Let A be a set of actions playable by the agent in the workplace reward maximizing game with $|A| \geq 2$. At time $t = 1, 2, \ldots, T$

(i) The agent plays a mixed strategy p^t, which is a probability distribution over A.

(ii) The adversary picks a utility (cost) vector $u^t : A \to [0, 1]$.[1]

(iii) An action u^t is then chosen according to the distribution p^t and the decision-maker incurs a cost of $u^t(a^t)$. The decision-maker learns the entire utility (cost) vector u^t and not just the realized cost $u^t(a^t)$.[2]

The actions of the workplace reward maximizing game is which colleague to work with. Hence in a workplace with n agents, there will be n possible actions for each agent; $n-1$ of them denote working with each of the other agent and the last being working by ownself. As in the definitions in no-regret dynamics we define regret and the no-regret algorithm the same way.

Definition 0.2 (Time-averaged regret)**.** The *time-averaged regret* of the action sequencce $a^1, a^2, \ldots, a^T$ with respect to a fixed action a is

$$\frac{1}{T}\sum_{t=1}^{T}\left[u^t(a) - u^t(a^t)\right]$$

Note: since the adversary will want to minimise your utility, the action that you play at each time t will give a smaller utility than playing a constant action.

Definition 0.3 (No-Regret Algorithm)**.** Let $\mathcal{A}$ be an online decision-making algorithm.

(a) The adversary for $\mathcal{A}$ is a function that takes as input the day t, the mixed strategies $p^1, p^2, \ldots, p^t$ produced by $\mathcal{A}$ on the first t days and the realized actions, $a^1, a^2, \ldots, a^{t-1}$ of the first $t-1$ days and produces as output a utility (cost) vector $u^t : A \to [0, 1]$.

[1] The other $k-1$ agents models the adversary well: when an agent i plays his strategy, other agents might play strategies that might possibly increase his cost which nicely simulates that the adversary picks a cost vector u^t after the agent i picks his mixed strategy.

[2] The learning of the entire cost vector is also reasonable in real-life scenarios as often after making a wrong decision, you will think of what if you played the other strategy that is more beneficial for you.

(b) An online decision-making algorithm has *no external regret* if for every adversary for it, the expected regret with respect to every action $a \in A$ is $\mathcal{O}(1)$ as $T \to \infty$.

The interpretation of the output of the utilty function must be explained more explicitly; the naive interpretation for now will be for a reward R_n obtained for a particular task n, the amount of reward shared with you will be $R_n(u^t(a_n))$ where a_n denotes the action of choosing task n.

The Algorithm

As we are measuring welfare using the agent's utility the *multiplicative weights* (MW) algorithm has to be modified accordingly for it to work with utility, that is the weights of each action increases as the rate of increase depends on the utility gained from an action.

1. Initialize $w^1(a) = 1$ for all $a \in A$
2. For $t = 1, 2, \ldots, T$
   (a) Choose a task to work on according to a distribution $p^t := w^t/\Gamma^t$, where $\Gamma^t = \sum_{a \in A} w^t(a)$ is the sum of the weights.
   (b) For a given utility vector, increase the weights using the formula $w^{t+1}(a) = w^t(a) \cdot (1+\epsilon)^{u^t(a)}$ for every action $a \in A$.

   Thus, if the proportion of the reward that is allocated to you is larger for working with a particular agent, then in the next time step, a larger weight is assigned for that agent.

0.1 Two Agent Scenario

Consisder two agents each with a project on its own. Let agent i has strategy p_{ij}^t: probability of joining project j. At each time step (which we shall more precisely define later), the project produces a reward R_j^t to be spilt among agents in that project such that

$$R_j^t = \sum_{i \in R} u_{ij}^t$$

Agent i update his weights with every time step such that for all i, j $\sum_i p_{ij}^t = 1$. This is done by the multiplicative weights (MW) update algorithm. For every time step the reward is calculated by the following in a two agent scenario

$$R_i(\mathbf{p})$$

and the agent's utility can be computed using the matrix

$$\begin{pmatrix} R_1 & R_2 \end{pmatrix} \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \end{pmatrix}$$

References

[Chakravarti et al., 2015] Chakravarti, A., He, C., and Wagman, L. (2015). Inducing knowledge sharing in teams through cost-efficient compensation schemes. *Knowledge Management Research & Practice*, 13(1)(2015):1–20.

[Drago and Turnbull, 1991] Drago, R. and Turnbull, G. K. (1991). Competition and cooperation in the workplace. *Journal of Economic Behavior and Organization*, 15(3):347–364.