POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Matematica
Scuola di Ingegneria Industriale e dell'Informazione

Solving Imperfect-Recall Games: New Representations and Algorithms.

Relatore: Prof. Nicola Gatti
Correlatore: Andrea Celli

Tesi di Laurea di:
Andrea Agostini
Matricola 862715

Anno Accademico 2017-2018

Sommario

In questa tesi studiamo

Abstract

This thesis focuses

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Area and Main Problem

The present work addresses problems.

1.2 Original Contributions

We start by focusing on s $\mathsf{P} = \mathsf{NP}$, even on a completely inflated tree. Then, we introduce the *hybrid column generation* algorithm which computes optimal team-maxmin equilibria with coordination devices (Celli and Gatti, 2018). We can use this

1.3 Thesis Structure

The Thesis is structured in the following way:

- In Chapter 4 we draw conclusions. We summarize the main results of our work and we propose some future developments.

Chapter 2

Problem Formulation

Standard MAB problem formulations...

2.1 The MAB Persistent problem

The Mab Persistent problem that we are going to analyze in this thesis is formalized in this section. Over a finite time horizon, composed of N time instants, at each time instant t, an agent must pull (choose) an arm (action) a_t from an arm set $A = \{1, \ldots, K\}$. When we pull an arm a_j at time t the environment returns a realization of a random variable $R_{j,t}$ and one of a random vector $\boldsymbol{Z_{j,t}} = (Z_{j,t,1}, \ldots, Z_{j,t,Tmax})$. The vector $\boldsymbol{Z_{j,t}}$ represtens the persistency of the feedback $R_{j,t}$, meaning that each component $Z_{j,t,m}$ describes which fraction of $R_{j,t}$ the agent will collect at the m-step. At each time instant from the pull of an arm, the learner will collect a reward called *instant reward* defined as follows:

Definition 1 (Instant Reward)**.** *We define the instant reward achived at time s, consequently to the pull of the arm j at time t as:*

$$r_{j,t,s} = R_{j,t} Z_{j,t,m}$$

Where $s \in \{t, \ldots, t + T_{max} - 1\}$ and $m = s - t + 1$

By setting the size of the vector $\boldsymbol{Z_{j,t}}$ to T_{max}, which is a fixed constant, we impose that the lasting of the feedback can be at most T_{max} steps. In according to the scenario we are dealing with, T_{max} will be known or not by the agent. We do not have any preliminary knowledge regarding the distributions of $R_{j,t}$ and $\boldsymbol{Z_{j,t}}$, we only know that $R_{j,t}$ has support in $[R_{min}, R_{max}]$ and $Z_{j,t,m}$ has support in $[0, 1]$ where $1 \leq j \leq K$, $1 \leq t \leq N$, $1 \leq m \leq T_{max}$. Each realization of $\boldsymbol{Z_{j,t}}$ is characterized by the number of steps that we have

to wait before having a positive component $Z_{j,t,m}$. We call this quantity *delay* and we formalize it in the following way:

Definition 2 (Delay). *We define the delay of a realization* $\boldsymbol{Z_{j,t}}$ *as:*

$$d_{j,t} = \sum_{m=0}^{T_{max}} \mathbb{1}_{\{Z_{j,t,m}=0 \ \wedge \ \forall k<m \ Z_{j,t,k}=0\}}$$

We are also interested in the position of the last positive component of a realization $\boldsymbol{Z_{j,t}}$. In essence, this quantity represents the true number of steps that we need to wait in order to collect all the instant rewards achievable after the pull of an arm. For this reason we call it *true length* and we define it as follows:

Definition 3 (True Length). *We define the true length of a realization* $\boldsymbol{Z_{j,t}}$ *as:*

$$l_{j,t} = \sum_{m=0}^{T_{max}} \mathbb{1}_{\{\exists k \ k\geq m \ | \ Z_{j,t,k}>0\}}$$

To better suit a variety of scenarios that require a persistence reward framework, we devise two distinct configurations:

- **General Persistency** We do not assume anything regarding $\boldsymbol{Z_{j,t}}$. This configuration turns to be suitable for scenarios in which the istant reward could be missing at a certain point and then reappear at a later time.
- **Tight Persistency** We impose that, giving a realization of a persistency vector, every non-zero component must be adjacent. More formally, we say that we are in *Tight Persistency* configuration if for each realization $\boldsymbol{Z_{j,t}} = (Z_{j,t,1}, \ldots, Z_{j,t,Tmax})$ it holds the following condition:

$$\nexists i, m, k \quad i < m < k \quad \text{s.t.} \quad Z_{j,t,i} > 0 \wedge Z_{j,t,m} = 0 \wedge \ Z_{j,t,k} > 0 \quad \forall i, m, k \in (1, \ldots, T_{max})$$

We now present a couple of examples derived from practical cases with the aim to highlight the needs that motivate the two configurations above mentioned.

Example 1. *We are the seller of an online magazine that works via subscription. In order to have access to our service, a new user can stipulate a contract with fixed duration and monthly fee. We let the possibility to suspend*

and restart the service in every moment during the contract simply stopping or making the monthly payments. Intuitively, we think that high prices discourage a continuous usage of the service while low prices could lead to stable subscriptions but with the risk of generating unsatisfactory profit. We are facing the problem of finding the best monthly price to assign at the service in order to maximize the revenue. This scenario can be directly modeled as a MAB persistent problem in ***General Persistency****. Each arm can be assigned to a specific fee designed as a valid option. When the agent pulls an arm the feedback extracted $R_{j,t}$ will be simply the price related to that arm. The persistency vector $\boldsymbol{Z_{j,t}}$, on the other side, will capture the adherence of the user to the service and will have a size of Tmax equal to the number of months of the contract. Every component of the vector will be a Bernoulli variable that takes the value 1 if the user has made the payment in a certain month, 0 on the contrary.*

Example 2. *We want to conduct an ethical clinical trial, In order to define which is the best medical treatment for a specific chronic illness. Let's say that we have two options available, a red pill and a blue one, hence we model them as two arms. Every day the agent must choose which one of the two therapies administer to a new patient on the basis of previous observations. Different from prior MAB application for this task, here we want to consider also the life quality of a patience in addition to his lifespan. For this reason after the treatment administration, a patient is tested every day and an index of his health status is computed. We can assume that this index is ranging from 0 to 1, where 1 represents a perfect state of health whereas 0 means that the patient is dead. This scenario could be easily addressed as a MAB persistent problem in* ***Thigh Persistency*** *configuration with delay steps equal to zero for each realization of the persistency vector. In fact, we can set T_{max} at the maximum lifespan possible after the diagnosis of the considered ilness, and we can model every component of the vector $\boldsymbol{Z_{j,t}}$ as the health status index. For this scenario $R_{j,t}$ could be fixed to a constant equal for each arm, letting the role of capturing the reward only to the persistency vector $\boldsymbol{Z_{j,t}}$. The Thigh Peristency condition holds, as a matter of fact, it does not make sense to have a positive index health status after a death.*

The nature of the presented problem lead us to introduce two definitions of reward achievable pulling an arm. In a straightforward manner, we call *Pull Reward* the the sum of the instant rewards gained thanks to the pull. In some scenarios could be reasonable to take in consideration also the time needed to collect all the instant rewards of a pull. Hence, we call *Normalized Pull Reward* the sum of instant rewards divided by the true length of the

persistency vector. Formal definitions are provided below.

Definition 4 (Pull Reward). *We define the pull reward achived pulling the arm j at time t as:*

$$X_{j,t} = \sum_{s=t}^{t+T_{max}-1} r_{j,t,s}$$

Definition 5 (Normalized Pull Reward). *We define the normalized pull reward achived pulling the arm j at time t as:*

$$Y_{j,t} = \frac{\sum_{s=t}^{t+T_{max}-1} r_{j,t,s}}{l_{j,t}}$$

For the sake of simplicity, we will refer to a generic reward of arm a_j at time t as $X_{j,t}$, in order to adopt the same notation of the standard MAB literature. However the definitions stated below will apply evenly to the *Pull reward* or the *Normalized pull Reward*, unless otherwise specified.
Successive plays of arm a_j yield rewards $X_{j,t_1}, X_{j,t_2}, \ldots$ which are random variables independent and identically distributed according to an unknown distribution with unknown expectation μ_j.

	Pull Reward	Normalized Pull Reward
General Persistency	example 1 Spostify Scenario	
Tight Persistency	example 2	Rent Scenario

2.1.1 Performance measures

The goal of a learning agent is to maximize its cumulated reward, the pulling strategy adopted to accomplish this task is referred as *policy*. In order to measure the performance of a policy, we compare its beahviour with the one of a fictitious algorithm, called *Oracle*, which constantly play the optimal arm. For this purpose we introduce de concept of *Regret*.

Definition 6 (Regret). *The Regret of a policy cumulated after n plays is definied as:*

$$R_n = \max_{j=\{1,\ldots,k\}} \sum_{t=1}^{n} X_{j,t} - \sum_{t=1}^{n} X_{a_t,t}$$

Where a_t is the arm played by the learner at time t and the first term $\max_{j=\{1,\ldots,k\}} \sum_{t=1}^{n} X_{j,t}$ represtents the reward cumulated by the Oracle up to time n.

Since both the rewards and the player's actions are stochastic, we introduce a form of average regret, called *pseudo-regret*.

Definition 7 (Pseudo-Regret)**.** *The Pseudo-Regret of a policy cumulated after n plays is definied as:*

$$\bar{R}_n = n\mu^* - \sum_{t=1}^{n} \mu_{a_t}$$

Where $\mu^* = \max_{j=\{1,\dots,k\}} \mu_j$ *is the expected reward of the optimal arm and* μ_{a_t} *is the expected reward of the arm played at time t.*

This form of regret is more suitable for the purpose of our analysis, therefore in the rest of thesis we will evaluate our algorithm in term of pseudo-regret and, for the sake of simplicity, from now on we will refer to it as regret.

Chapter 3

Novel Algorithms

3.1 Grounding Concepts

Multi-armed Bandit problems pertain to Online Learning, which means that in order to evaluate algorithms we need to design a simulation environment where we can mimic real-scenarios dynamics. From a practical perspective we need a data structure which represents the concept of persistent reward. In order to do that we define a vector called *Bucket* aimed to contain the rewards obtained at each time steps. With $r_{b,m}$ we refer to the reward at position m on bucket b. The *Bucket* has a fixed size *Tmax*. During the simulation, the bucket will be parsed according to the experiment time, meaning that the learner can only visit the bucket-cells containing the rewards of the past. When at time t an arm a_j is played, the environment generates a bucket $b_{t,j}$ that is collected by the learner. We call B_j the set of buckets collected after playing a_j. The algorithms described below have a structural difference from the standard ones designed for traditional Multi-armed Bandit, infact in the presented framework more than one arms can have a set of active buckets (not totally parsed yet) simultaneously. This parallelism implies that at each time instant we need to update the terms of every arms and not only of the last one played.

3.2 Baseline UCB

The pseudo-code of the Baseline UCB is depicted in Algorithm1. This algorithm extends the idea of the well known UCB algorithm in the case of persistent rewards. The reward is considered as a unique variable available after the termination of the bucket. The first k istants form the initialization phase, during which each arm is chosen once. After the k-th time istant the

loop phase begins. Here the agent plays the arm having the largest index $u_j(t)$, the upper confidence bound of the arm a_j at time t. It is the sum of the empirical reward of the arm measured so far $\hat{\mu}_j(t)$ and the exploration term $c_j(t)$. After the play of an arm, the *Update Procedure* occurs. For each arm it scans every bucket collected and when a terminated bucket is detected (after $Tmax$ istants from his generation) a new reward is observed. $T_j(t)$ representes the number of times arm a_j has been pulled up to time t. R_j is the reward of the arm a_j (SISTEMARE).

Algorithm 1 `Baseline UCB`

Require: arm set $A = \{a_1, a_2, \ldots, a_k\}$, time horizon N, bucket size $Tmax$

function POLICY
  for $t \in \{1, \ldots, k\}$ **do** ▷ init phase
    play arm a_t
    UPDATE
  end for
  for $t \in \{k+1, \ldots, N\}$ **do** ▷ loop phase
    play arm a_z such that $z = argmax_j\ u_j(t)$
    UPDATE
  end for
end function

function UPDATE
  for each arm $a_j \in A$ **do**
    for each bucket $b \in B_j$ **do**
      if b is ended **then**
        $x \leftarrow R_j * \sum_{m=1}^{Tmax} r_{b,m}$
        $\hat{\mu}_j(t) \leftarrow \frac{\hat{\mu}_j(t-1)*(T_j(t)-1)+x}{T_j(t)}$
        $B_j \leftarrow B_j \setminus \{b\}$
      end if
    end for
    $c_j(t) \leftarrow R_j * Tmax * \sqrt{\frac{2log(t)}{T_j(t)}}$
    $u_j(t) \leftarrow \hat{\mu}_j(t) + c_j(t)$
  end for
end function

3.3 Bound1

- $v_{j,m}(t)$ is the number of time an arm a_j visited a bucket at istant m up to time t

Algorithm 2 `Bound1`

Require: arm set $A = \{a_1, a_2, \ldots, a_k\}$, time horizon N, bucket size $Tmax$

function POLICY
  for $t \in \{1, \ldots, k\}$ **do** ▷ init phase
    play arm a_t
    UPDATE
  end for
  for $t \in \{k+1, \ldots, N\}$ **do** ▷ loop phase
    play arm a_z such that $z = argmax_j\ u_j(t)$
    UPDATE
  end for
end function

function UPDATE
  for each arm $a_j \in A$ **do**
    for $m \in 1, \ldots, Tmax$ **do**
      $\hat{\mu}_{j,m}(t) \leftarrow (\sum_{b \in B_j\ visistable\ up\ to\ m} r_{b,m})/v_{j,m}(t)$
      $c_{j,m}(t) \leftarrow \sqrt{\frac{2log(t * Tmax^{\frac{1}{4}})}{v_{j,m}(t)}}$
    end for
    $u_j(t) \leftarrow R_j * \sum_{m=1}^{Tmax} min(1, \hat{\mu}_{j,m}(t) + c_{j,m}(t))$
  end for
end function

Chapter 4

Conclusions

In this work, we studied the relationship between games with imperfect recall and team games. First, we described how to apply the inflation operation in practice. Computing a completely inflated tree requires only polynomial time, but a completely inflated tree may still have imperfect recall. Then, we defined personalities as a way to decompose a player with imperfect recall in a set of team members with perfect recall. Specifically, we defined an auxiliary team game

Bibliography

A. Celli and N. Gatti. Computational results for extensive-form adversarial team games. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.