DIVISION OF LABOUR IN COMMUNICATING MULTI-ARMED BANDITS

Preetham Venkatesh
Bhaskar Kumawat

Mid-term Project Report

Abstract

In this report we introduce the classical multi-armed bandit problem and the algorithms that have been designed in previous works to allow an agent to achieve minimal regret. We then describe a deterministic MABP that involves communication and information sharing among multiple MABs. In a variety of biological settings, information sharing (or equivalently the sharing of public goods) has been shown to bring about emergence of division of labour among multiple agents. Our goal is to design a mechanism or a generalized set of mechanisms where division of labour is the final outcome given each MAB is intelligent and rational. This is further supplemented with a road-map for future work including - spatial localization of certain specialized agents given a heterogeneous environment, and the differentiation of agents into 'demes' that share information locally.

1 INTRODUCTION

The multi-armed bandit (MAB) is a classic problem in game theory, finding applications in diverse fields such as crowdsourcing, online advertising [1], as well as power supply [2] and funding allocation [3]. The name is derived from its roots in gambling, where a player is required to pull one of K independent arms for multiple time periods. An arm, when pulled gives a reward which the player seeks to learn while simultaneously trying to maximize the reward he gets over the course of the game. It thus represents the trade-off between exploration (trying a new arm that might yield better results) and exploitation (stick to your best performing arm) which forms the basis of reinforcement learning. Regret in the game is defined as the deviation from the maximum reward that can be achieved in the game, and thus regret minimization represents another perspective of the problem. There are several variations of the game that have been analyzed over the years. Two popular ones have been the contextual bandit [4], where the player receives a context vector associated with each iteration, and the player thus seeks to correlate the context with the reward distribution, and the adversarial bandit [13], where player 1 pulls an arm and player 2 decides the reward distribution of each arm simultaneously. We shall restrict our focus to stochastic MAB for the next two sections.

2 MULTI ARMED BANDIT PROBLEM

The multi-armed bandit problem is formalized as follows: Let N be the number of arms, t denote a time step, x_i^t denote the reward associated with pulling arm i at time step t. Let us assume the rewards are bounded, say $x_i^t \in [0,1]^n$. The MAB game then goes as follows:
For t=1,2,3..T:

- Player selects an arm i
- Simultaneously, the environment selects the reward vector $x^t = (x_i^t....x_i^t)$ in $[0,1]^n$
- The reward $x_{i_t}^t$ is observed

A MAB algorithm α takes in the sequence of the arms pulled $(i_1, x_{i_1}^1), (i_2, x_{i_2}^2), ..., (i_{t-1}, x_{i_{t-1}}^{t-1})$ and returns an arm $i_t \in [n]$ to pull on trial t. The reward of algorithm α on a sequence of reward vectors $(x^1, ..., x^T)$ is given by:

$$G_T[\alpha] \equiv G_(x^1, ..., x^T)[\alpha] = \Sigma_{t=1}^T x_{i_t}^t$$

The reward of a fixed arm i on a sequence of $(x^1, ..., x^T)$ is given by:

$$G_T[i] \equiv G_(x^1, ..., x^T)[i] = \Sigma_{t=1}^{T} x_i^t$$

The regret of an algorithm α over the sequence $(x^1, ..., x^T)$ is given by:

$$R_T[\alpha] \equiv R_(x^1, ..., x^T)[\alpha] = max_{i \in [n]} G_(x^1, ..., x^T)[i] - G_(x^1, ..., x^T)[\alpha]$$

3 ALGORITHMS

For the stochastic MAB problem, there are several algorithms that achieve sublinear regret [10]. We shall briefly describe each of them. 1) Exploration separated algorithms: All arms are pulled sequentially for a pre-decided number of rounds, followed by repeated pulling of the best arm. 2) Upper Confidence Bound (UCB) algorithms: Here, each arm is assigned an index that depends on its performance in the previous rounds and a constant which represents the trade-off. 3) Bayesian algorithms: A prior distribution is assumed for each of the arms. This distribution is updated with each passing round. UCB can be implemented in many ways, such as UCB1 [6], UCB-Normal, (α,ψ) UCB algorithm [7], KL-UCB etc. Similarly, Thompson sampling, where rewards follow a Bernoulli distribution, and the Bayes-UCB algorithm [9] are two ways to implement Bayes algorithms.

4 ROAD-MAP

4.1 The MAB communication problem

We start by introducing a **network** of rational multi-armed bandits that learn over time in order to maximize their overall utilities. The individual agents play separate deterministic MAB games where the information obtained from pulling an arm is fixed. The information can also be obtained from immediate neighbours of an agent and this ultimately presents an exploration vs. exploitation trade-off as is seen in classical MAB problems.

This acquisition of information happens at every time-step and the overall information that an agent has about each arm is translated to utility at the end of the time-step through a set of utility functions (one for each arm). The only knowledge that persists over time-steps (and promotes learning) is the set of instructions executed by an agent in the previous time-step and the utility obtained. This setup closely mirrors the public goods scenario seen in multiple biological systems where reaction intermediates to metabolic processes can be shared among cells [8]. It has been shown that cells tend to evolve to share these intermediates, especially in cases where the initial reaction substrate is scarce [5]. A more formal statement of the problem is as follows.

4.2 Problem setup

Consider a set of players $\mathbb{N}$ each with a set of arms $\mathbb{K}$ available to pull at each time step.

$$\mathbb{N} = \{1, 2, 3, ..., N\}$$

$$\mathbb{K} = \{1, 2, 3, ..., K\}$$

mnv c

M_n refers to the set of neighbours of a player $n \in \mathbb{N}$. For simplicity and ease of visualization we will limit our discussion to networks where each player is a point on a 2 dimensional grid and can only communicate with its four immediate neighbours. In the figure below, $M_5 = \{2, 4, 6, 8\}$.

Each agent can play only perform a limited number of operations at a time-step bounded by η - the computational power available to each player per time-step. An operation can either be pulling of an arm, sending of information or receiving of information.

Because of this computational cutoff we also divide each time-step into η smaller units. Each successive instruction in a time step is executed simultaneously for all agents in the network. The array of instructions to be executed in a time step thus act as a 'genome' for the agent. This 'genome' can be modified over time-steps to increase total utility. A time-step is thus the evolutionary time-scale over which the 'genome' of the agent can change.

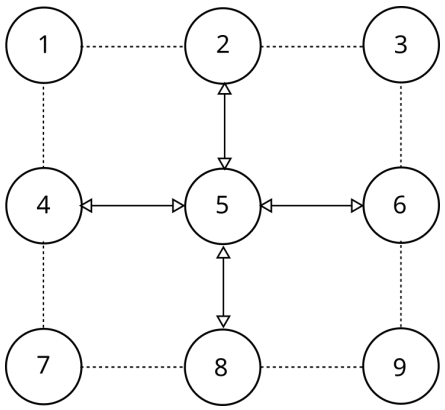

Figure 1: An example network of MAB agents

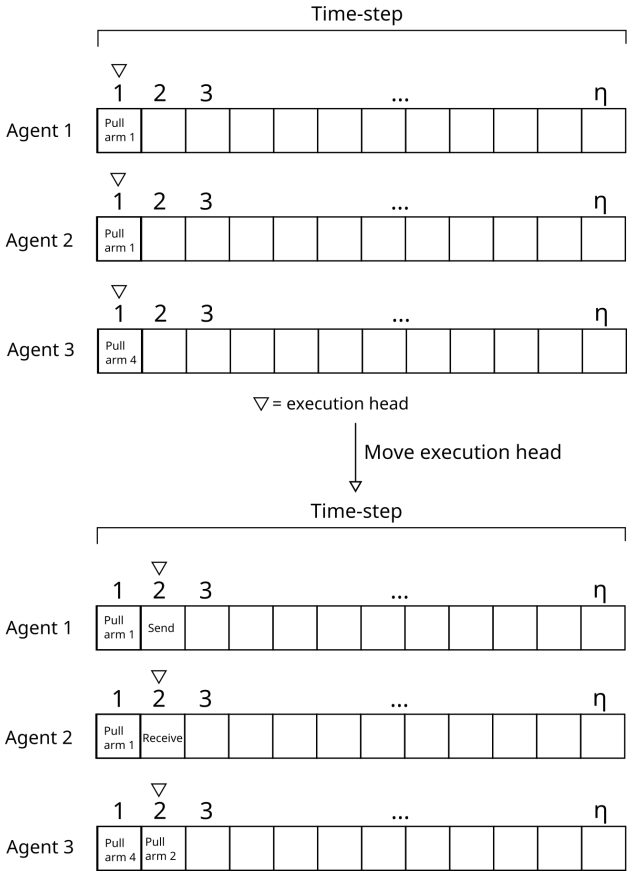

Figure 2: An agent can execute a total of η instructions in a time step.

4.3 Procedure

At each time-step, every player n executes the instructions in its 'genome' in the order they are specified therein. The instructions can be one of the following.

1. **Pull** a set of arms to specify a distribution $S_n(k)$ that indicates the number of times an arm $k \in \mathbb{K}$ has been pulled by the player.

   All pulling of arms is done at the start of the player's genome before any other instructions are executed. The information gets added to the player's grand information distribution $I_n(k)$(defined in next section) . Hence, $I_n(k) \rightarrow I_n(k) + S_n(k)$

2. **Transmit** the information from $\psi_n(k)$ out of $S_n(k)$ pulls for an arm $k \in \mathbb{K}$.

   The player can only transmit an amount of information (about arm k) less than or equal to what it obtained by pulling the arm k. Hence,

   $$\psi_n(k) \leq S_n(k), \forall k \in \mathbb{K}$$

   This information is lost from the information available to the player. Hence, $I_n(k) \rightarrow I_n(k) - \psi_n(k)$

3. **Receive** information about $\rho_n(k)$ pulls of arm k from the neighbours.

   As $\rho_n(k)$ is just the sum of information received from all neighbours in M_n it can be written as a sum of the information transmitted by neighbours. An integral factor comes in because of equipartition of the information among immediate neighbours.

   $$\rho_n(k) = \frac{\sum_{j \in M_n} \psi_j(k)}{4}$$

   This information is added to the information available to the player. $I_n(k) \rightarrow I_n(k) + \rho_n(k)$

At the end of genome execution, the agent is rewarded with a total utility as described in the next section. This reward is calculated from the total information available with the player at the end of execution. As the number of instructions is limited by computational power/genome-size η.

$$\sum_{k=1}^{K} S_n(k) + \text{no. of times information is transmitted}$$
$$+\text{no. of times information is received} = \eta$$

4.4 Utility

The utility in this problem is a function of the total information available to the player about all the arms. We define a grand information distribution as follows.
Define: Grand information distribution I_n for player n.

$$I_n(k) = S_n(k) - \sum \psi_n(k) + \sum \rho_n(k)$$

The summations on ψ_n and ρ_n are over the total number of send and receive instructions in the genome.

Associated with each arm k is a utility function U_k that takes in the grand information distribution of the player n and rewards it a utility for that arm at the end of genome execution. The total utility gained by the player in the time-step is,

$$\sum_k U_k(I_n)$$

Note that as the utility from an arm is a function of the whole information *distribution* and not the

information about a particular arm k, there can be added advantage (or disadvantage) of having knowledge about arms other than the one for which the utility is being calculated.

4.5 Learning

The learning happens over time steps as the 'genomes' of the agents change. We can define a context in which the instructions at each timestep are chosen. This context is just a vector consisting of the instructions that were played at the last time step and the reward obtained from those instructions.

Algorithms can be designed to use this context vector in order to maximize utility - this algorithm determines the exploration vs. exploitation priority as mentioned before. An agent can either decide to use its previous genome or modify it and see if an increase in utility is observed.

4.6 Mechanism Design

The goal is to design a mechanism Ω,

$$\Omega =< U_1(I), U_2(I), U_3(I), ..., U_K(I) >$$

that prescribes a set of utility functions which make the agents co-operate and share information about arm pulls with each other. We believe that a set of mechanisms will also lead to agents specializing in pulling certain arms so as to maximize their overall utility.

While the rewards from the arms are deterministic, the inability to predict the actions of neighbouring agents makes this problem non-trivial. For example, if a neighbour transmits information about a highly rewarding arm, it might be favourable to receive that information rather than pull the arm yourself. The upper bound on genome-size/computational power makes the choosing of instructions for an agent critical.

4.7 Future work

Apart from the above assumptions, the following additional features can be incorporated into the problem.

- Spatial heterogeneity in the information attained from each arm. This can in-principle lead to stronger specialization in the kind of arms pulled by bandits depending on their position in the network. [12]
- Implementation of a unequal sharing function that allows the players to share information only with some neighbours. This can lead to the formation of demes, which are groups of closely related agents that communicate minimally outside the deme.

References

[1] H. Nazerzadeh, A. Saberi and R. Vohra, Dynamic pay-per-action mechanisms and applications to online advertising, Operations Research, 61(1) (2013), 98-111

[2] S. Jain, B. Narayanaswamy and Y. Narahari, A multiarmed bandit incentive mechanism for crowdsourcing demand response in smart grids, In Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27-31, 2014, Quebec City, Qu ebec, Canada , (2014), 721-727.

[3] Gittins, J. C. (1989), Multi-armed bandit allocation indices, Wiley-Interscience Series in Systems and Optimization., Chichester: John Wiley Sons, Ltd.,

[4] Langford, John; Zhang, Tong (2008), "The Epoch-Greedy Algorithm for Contextual Multi-armed Bandits", Advances in Neural Information Processing Systems 20, Curran Associates, Inc., pp. 817824

[5] Koschwanez, John H., Kevin R. Foster, and Andrew W. Murray. "Improved use of a public good selects for the evolution of undifferentiated multicellularity." Elife 2 (2013): e00367.

[6] P. Auer, N. Cesa-Bianchi and P. Fischer, Finite-time analysis of the multiarmed bandit problem, Journal of Machine Learning, 47(2-3) (2002), 235-256.

[7] S. Bubeck and N. Cesa-Bianchi, Regret analysis of stochastic and nonstochastic multi-armed bandit problems, Foundations and Trends in Machine Learning, 5(1) (2012), 1-122.

[8] Allen, Benjamin, Jeff Gore, and Martin A. Nowak. "Spatial dilemmas of diffusible public goods." Elife 2 (2013): e01169.

[9] E. Kaufmann, N. Korda and R. Munos, Thompson sampling: An asymptotically optimal finite-time analysis, In Algorithmic Learning Theory - 23rd International Conference, ALT 2012, Lyon, France, October 29-31, 2012. Proceedings, (2012), 199-213.

[10] Jain, Shweta, et al. "Mechanisms with learning for stochastic multi-armed bandit problems." Indian Journal of Pure and Applied Mathematics 47.2 (2016): 229-272.

[11] `http://www.shivani-agarwal.net/Teaching/E0370/Aug-2013/Lectures/22.pdf`

[12] Furusawa, Chikara, and Kunihiko Kaneko. "Origin of multicellular organisms as an inevitable consequence of dynamical systems." The Anatomical Record: An Official Publication of the American Association of Anatomists 268.3 (2002): 327-342.

[13] Auer, Peter, et al. "Gambling in a rigged casino: The adversarial multi-armed bandit problem." focs. IEEE, 1995.