Students, Exams and Bandits

by Prabhat Lankireddy (22M2156) and Rahul Choudhary (200070065)

EE6106 Online Learning and Optimization - Course Project
IIT Bombay

Abstract

What is the best strategy a student can follow when writing an exam to get maximum possible marks? Can we obtain an answer to this question by thinking of exam writing as a multi-armed bandit problem? We intend to explore these questions in our project. We pose the exam writing scenario as a multi armed bandit problem with stochastic time consumption, where each question takes a variable number of time steps to solve. Each question in the exam is an arm with a fixed reward (marks awarded in answering the question) and the student (the agent implementing an algorithm) must choose the optimal order of questions in order to maximize the overall reward within a fixed time horizon T. We analyze this simple setup and discuss a more complicated version of the problem later on.

1 Introduction

As many things in life, writing an exam is a sequential decision making problem. The student is given a list of questions worth some amount of marks, and is expected to solve the question paper within an allotted amount of time, with the goal of obtaining as many marks as possible overall. Because different students have spent different amounts of time studying different topics from the exam syllabus, no two students perceive the exam paper in the exact same manner. A question that might be perceived as hard for one student might be easy for another. Given such a problem setting, can we come up with a general strategy that a student use to obtain maximum possible marks in any exam?

In this project, we try to model the sequential decision making problem of exam writing as a multi-armed bandit problem with budget constraints, with the student being the decision making agent and the questions being the arms. Note that the student only knows the maximum marks that can be given for an answer, but does not exactly know how many marks will be given to his/her answer; the student can only estimate how many marks will be given for a particular written answer. For this reason, the rewards are in fact *assumed rewards* and are taken to be stochastic and bounded in the problem setting. Moreover, each question takes an arbitrary amount of time to solve, and the student is expected to finish the exam in some allotted time. We include this characteristic of the problem in the problem setting by asssuming that each arm takes a random number of timesteps to solve, and the agent must work with a certain limited number of timesteps and maximize reward. The final constraint is that one question can be answered only once. For learning to happen, there must be something that the student learns from answering one question that can be used in the next question. This *shared information* is modeled differently in different, more particular, problem settings as discussed below.

From the above general problem setting, we first consider a simpler setting in which the students are assumed to be *perfect* i.e. the student always obtains full marks to any given answer. We also assume that the expected time taken to solve the question is proportional to the maximum marks that can be obtained by answering a question. This is the *shared information* among arms in this setting, as discussed above. In this case, the problem boils down to coming up with an algorithm that tells the student which question to answer next to maximize expected reward, given the history of answered questions and the time left.

Then we briefly discuss a more complicated setting in which the students perceived a few questions as easy and the others as difficult, which makes the rewards stochastic. We also assume that the students can stop writing an answer if it takes more time that expected, so that the student can instead attempt other questions and maximize marks. In this setting, a decision making agent must decide two things: the arm to be pulled and the expected time to write an answer. If the student takes more time to write an answer than the expected time, then the question is skipped and the student moves on to the next question. We cast this into the *Farewell to Arms (F2A)*[1] setting and show that the Wait-UCB algorithm that they propose can work well if we consider baskets of similar questions as a single *macro arm* and the expected time delay to answer a question from a basket as the *micro arm*. We also consider a variation of the Wait-UCB algorithm by using the notion of *shared information* as discussed above and empirically compare results with Wait-UCB.

2 Perfect students with stochastic time consumption

We develop the problem setting while keeping the optimal exam-writing problem in mind. Let ν be a multi armed bandit instance (exam paper) with K arms (questions). For each arm $i \in \{1, 2, ..., K\}$, there is a corresponding maximum reward r_i. However, this reward is obtained after a stochastic time t_i, which can take the value of any positive integer. Let T be the total number of timesteps available to the agent as budget, and let $T_i = \mathbb{E}(t_i)$. Without loss of generality, we consider the reward ordering $r_1 \geq r_2 \geq \cdots \geq r_K$. We expect the arms associated with higher rewards to take more time, so we have the following for all integers i, j such that $i < j$

$$\mathbb{P}(t_i \leq t) \leq \mathbb{P}(t_j \leq t) \; \forall t \in \mathbb{N} \tag{1}$$

Note that this gives us the ordering $T_1 \geq T_2 \geq \cdots \geq T_K$.

2.1 Exam with 2 questions

Say there are only 2 questions in the exam paper. Which question should the student answer first? To answer this question, let's consider the case of $K = 2$ where $r_1 > r_2$. We have 2 policies π_1 and π_2 such that the policy π_1 chooses to pull arm 1 before arm 2, while policy π_2 chooses the opposite. The expected reward of policy π_1 is given by

$$\begin{aligned}
\mathbb{E}_{\pi_1}(\text{Reward}) &= \mathbb{E}(r_1 \mathbb{I}\{t_1 \leq T\} + r_2 \mathbb{I}\{t_1 + t_2 \leq T\}) \\
&= r_1 \mathbb{E}(\mathbb{I}\{t_1 \leq T\}) + r_2 \mathbb{E}(\mathbb{I}\{t_1 + t_2 \leq T\}) \\
&= r_1 \mathbb{P}(t_1 \leq T) + r_2 \mathbb{P}(t_1 + t_2 \leq T)
\end{aligned}$$

Similarly, we get $\mathbb{E}_{\pi_2}(\text{Reward}) = r_2 \mathbb{P}(t_2 \leq T) + r_1 \mathbb{P}(t_1 + t_2 \leq T)$. Using this, we express the difference of expected rewards between both policies as follows.

$$\begin{aligned}
\mathbb{E}_{\pi_1}(\text{Reward}) - \mathbb{E}_{\pi_2}(\text{Reward}) &= r_1 \mathbb{P}(t_1 \leq T) + r_2 \mathbb{P}(t_1 + t_2 \leq T) - r_2 \mathbb{P}(t_2 \leq T) + r_1 \mathbb{P}(t_1 + t_2 \leq T) \\
&= r_1 \mathbb{P}(t_1 \leq T) - r_2 \mathbb{P}(t_2 \leq T) - (r_1 - r_2) \mathbb{P}(t_1 + t_2 \leq T)
\end{aligned}$$

2.1.1 Case: All questions are expected to be solved within time T

In this case, we have $\mathbb{P}(t_1 \leq T) \approx \mathbb{P}(t_2 \leq T)$. We obtain the following inequality using the above expression.

$$\begin{aligned}
\mathbb{E}_{\pi_1}(\text{Reward}) - \mathbb{E}_{\pi_2}(\text{Reward}) &= r_1 \mathbb{P}(t_1 \leq T) - r_2 \mathbb{P}(t_2 \leq T) - (r_1 - r_2) \mathbb{P}(t_1 + t_2 \leq T) \\
&\approx (r_1 - r_2) \mathbb{P}(t_1 \leq T) - (r_1 - r_2) \mathbb{P}(t_1 + t_2 \leq T) \\
&= (r_1 - r_2)(\mathbb{P}(t_1 \leq T) - \mathbb{P}(t_1 + t_2 \leq T)) \\
&\geq 0
\end{aligned}$$

The first inequality arises from $\mathbb{P}(t_1 \leq T) \approx \mathbb{P}(t_2 \leq T)$ and the last inequality comes because $r_1 - r_2 \geq 0$ and $\mathbb{P}(t_1 \leq T) - \mathbb{P}(t_1 + t_2 \leq T) \geq 0$. This tells us that solving the question with the higher marks first is better.

2.1.2 Case: Question 1 takes too much time to solve

In this case, we have $\mathbb{P}(t_1 \leq T) \approx \mathbb{P}(t_1 + t_2 \leq T)$. We obtain the following inequality using the above expression.

$$\begin{aligned}\mathbb{E}_{\pi_1}(\text{Reward}) - \mathbb{E}_{\pi_2}(\text{Reward}) &= r_1 \mathbb{P}(t_1 \leq T) - r_2 \mathbb{P}(t_2 \leq T) - (r_1 - r_2)\mathbb{P}(t_1 + t_2 \leq T) \\ &\approx r_2(\mathbb{P}(t_1 + t_2 \leq T) - \mathbb{P}(t_2 \leq T)) \\ &\leq 0\end{aligned}$$

This tells us that in the case where hard questions take too much time to solve, it's better to solve the easier questions first in order to get more marks.

2.2 Algorithm for K questions

Inspired by the above analysis, we propose the following algorithm.

Algorithm 1

Initialize T, $S_r \leftarrow 0$, $S_t \leftarrow 0$.
Pull arm k with least reward r_k and obtain time consumed $\tilde{t}_k$.
$T \leftarrow T - \tilde{t}_k$
Initialize $a \leftarrow \frac{\tilde{t}_k}{r_k}$.
Until all questions are solved:
  Calculate $\hat{T} = \sum a\, r_i$, where summation is over all remaining arms.
  If $\hat{T} \leq T$
    Play arm with highest reward and obtain time consumed $\tilde{t}$
  Else
    Play arm with least reward and obtain time consumed $\tilde{t}$
  $a \leftarrow \frac{\text{Sum of all times obtained till now}}{\text{Sum of all rewards obtained till now}}$
  $T \leftarrow T - \tilde{t}$

The idea behind this algorithm is simple: estimate whether all the remaining questions can be solved on time. If they can be solved in time, then answer the question with the highest marks, else answer the question wiith the least marks. The algorithm assumes a linear relation between the marks allotted to an question and the expected time taken to solve that question, which gives the expression

$$\mathbb{E}(t_i) = T_i = a\, r_i \tag{2}$$

where $i \in [K]$.

3 Experiments and Results

We test our algorithm against the following three algorithms:

1. *Least marks first*: Always answer the question with least marks first.
2. *Highest marks first:* Always answer the question with highest marks first.

3. *Random*: Choose a random question to answer.

We test these algorithms on a 100-mark question paper with questions of marks $\{20, 15, 12, 11, 10, 9, 8, 5, 4, 3, 2, 1\}$. The agent has a time budget of $T = 180$ minutes, and each question takes a random number (integer) of minutes to solve. We assume that the stochastic time delays are Poisson distributed. We test these algorithms on different time distributions that have different Poisson parameters λ. The following table contains the total marks obtained by each of the algorithms in the presence of time delay distributions with different parameters.

Algorithm	**Poisson Parameters**				
	$\lambda = b - 0.1$	$\lambda = b$	$\lambda = b + 0.5$	$\lambda = b + 1$	$\lambda = b + 2$
Our Algorithm	**100.00**	89.15	71.30	**59.95**	**42.75**
Least marks first	92.00	87.00	69.50	59.00	41.50
Highest marks first	**100.00**	**96.70**	**72.80**	58.30	41.60
Random	97.20	88.20	72.60	58.35	41.95

Table 1. Total marks obtained by algorithms under different time delay distributions. Note that $b = \frac{T}{\sum r_i}$ is the ratio of available time budget and sum of all rewards.

We observe that the proposed algorithm performs well in cases where λ significantly varies from the ratio of total budget and total reward. The following figures show how the algorithm obtains rewards over time under different settings.

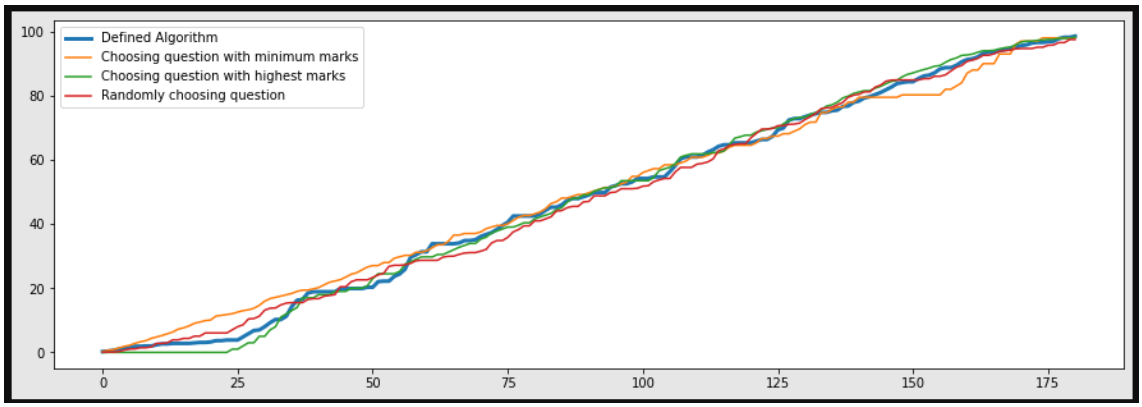

Figure 1. This figure shows how different algorithms obtain rewards over time, when delays are Poisson distributed with parameter $\lambda = b - 0.1$. Averaged over 20 runs.

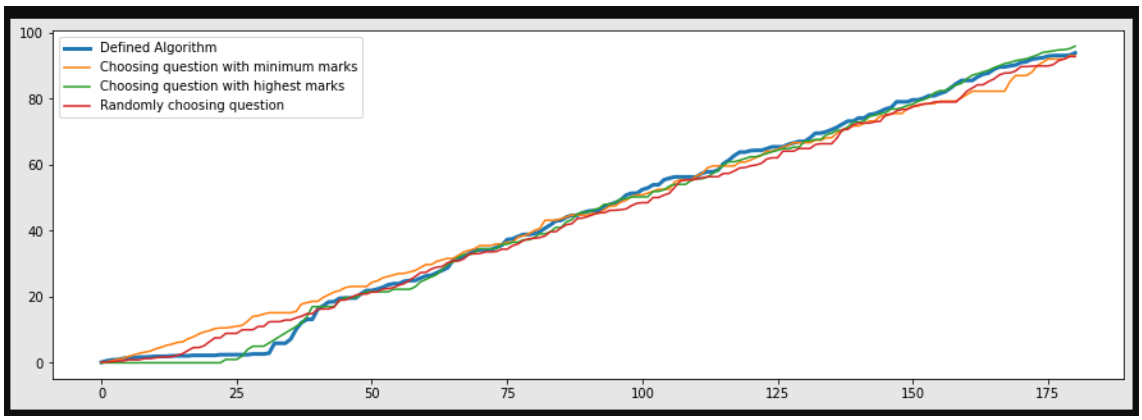

Figure 2. This figure shows how different algorithms obtain rewards over time, when delays are Poisson distributed with parameter $\lambda = b$. Averaged over 20 runs.

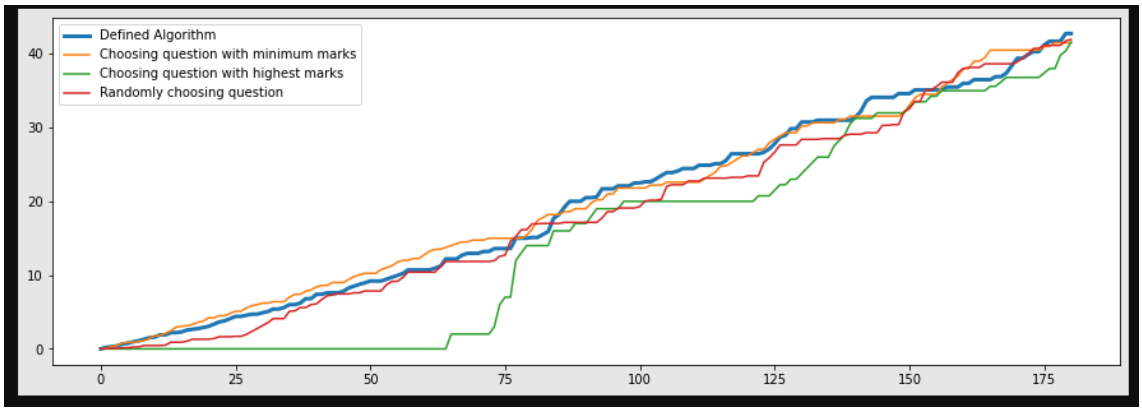

Figure 3. This figure shows how different algorithms obtain rewards over time, when delays are Poisson distributed with parameter $\lambda = b + 2$. Averaged over 20 runs.

4 Stochastic rewards with stopping times

We can consider a more complex problem setting in which the student is not perfect, and is willing to abandon a question if it's taking too much time to solve it. This problem can be set into the *Farewell to Arms* problem setting [1] in which the agent must choose a macro arm i (question) along with a micro arm j (delay). Then the reward is expressed by,

$$\text{Reward}(i, j) = X_i \mathbb{I}\{\tau < j\} \tag{3}$$

where X_i is the stochastic reward obtained from pulling arm i, and τ is the stochastic delay incurred after pulling arm i. If the delay τ is smaller than the micro arm, then the reward X_i is obtained, else the agent pulls another arm.

The algorithm proposed by [1], called the Wait-UCB algorithm, uses the ratio of rewards to times to calculate the UCB index. This idea blends in well with the linear model between rewards and time that we previously considered. However, any macro-micro arm pair can be considered any number of times in the problem setting assumed by [1], while that cannot be done in our case because a question cannot be solved once it has already been solved.

However, we can consider a *baskets* of questions and consider each basket as a macro arm. To integrate our previously assumed linear model, we can create these baskets of questions based on two factors: marks awarded to the question and the perceived difficulty of the question by the student. Higher marks and higher perceived difficulty leads to more time consumption while lower marks and lower perceived difficulty leads to lower time consumption. Using this concept of baskets of questions, we can adapt and improve the Wait-UCB algorithm to our problem setting, possibly leading to better results. This is an idea for future work.

5 Conclusion

In this project, we consider a multi-armed bandit problem with stochastic time consumption and come up with an algorithm to maximize expected reward in such a setting. We empirically run experiments on the algorithm and compare it with a randomized algorithm to analyse its performance. As an idea for future work, we also discuss a more complex problem setting in which one can choose a stopping time beyond which the agent would choose to pull another arm. A UCB-type algorithm can be developed in that case, which uses the ratio of rewards to times as an index. We thank Prof. Manjunath and Prof. Jayakrishnan Nair for giving us the opportunity to do this project and introducing us to this fascinating world of online learning!

Bibliography

[1] P Sharoff, Nishant A Mehta, and Ravi Ganti. A Farewell to Arms: Sequential Reward Maximization on a Budget with a Giving Up Option.