6.895 Theory of Parallel Systems Lecture 22

Pick a Winner

Lecturer: Tom Leighton *Scribe: He Yuxiong and Sajindra Jayasena*

Lecture Summary

1. *The Pick-a-Winner game*
   The section describes the rules of the pick-a-winner game and shows two simple observations.

2. *Strategies to maximize the profit*
   This section describes the optimal strategy, analyzes its performance, shows the key intuitions and sketches the proof of the performance guarantee theorem.

3. *Pick-a-Winner - K player Game*
   This section looks at how the game and strategy vary with multiple players

4. *Applications*
   This section looks at some applications that can be modelled using Pick-a-Winner single player and multi-player games

In the lecture, we consider a class of problems in which a decision-maker is required to select from among numerous dividend-paying options. The goal of the decision-maker is to select an option (or a set of options) that will have the best future return. The goal of the lecture is to show how to use probabilistic method to get the performance guarantee without making probabilistic assumptions.

1 The Pick-a-Winner game

The section describes the rules of the pick-a-winner game and shows two simple observations.

Pick-a-winner game has one player. The player is required to select one from among n dividend-paying options. The selection can be made on any day within d days. Once the player makes the decision, he is not allowed to change it. The profit of the player is equal to the dividends paid by the option after it has been selected. The goal of the game is to maximize the profit.

In the online version of this game, the player only knows the past dividend payments but he does not know the future payment schedule. In addition, the past and the future schedule may not be correlated. In fact, the future is decided by the adversary who knows the player's strategy and seeks to minimize the profit of the player. The question is how much money the player can make.

We can make two simple observations based on the rules of the game.

Observation 1 *The profit for any deterministic player is 0, even if* $P_{opt} = \$d$.

Proof Before the player makes any decision, the adversary pays a dividend for every option on every day. After the player makes a selection, the dividends are stopped for the selected option. The adversary can still pay a dividend for all the other options on every day. So even if $P_{opt} = \$d$, the profit of the deterministic player is 0. $\square$

Observation 2 *By choosing a random option at the start, we can get an expected profit* P_{opt} */n.*

Proof Since the adversary only knows our strategy but not the random number, we have a $1/n$ chance of selecting the optimal option. $\square$

2 Strategies to maximize the profit

This section describes an optimal strategy, analyzes its performance, shows the key intuitions and sketches the proof of the performance guarantee theorem.

An **Optimal Strategy** is described below:
Player selects target profit P and confidence level $1-q$. We label the options as $O_1, O_2 \dots O_n$. Each day, the player checks the status of each option to see which one has a dividend. For each x ($x = 1 \dots n$), if O_x has a dividend on the day, the player flips a coin to decide whether or not to make a selection to O_x . The player will make a selection on O_x with probability $2^{(qi/P)}/(Pn^2)$, where i is the total number of dividend O_x has generated so far. If the selection has been made, the player stops flipping coins and waits for the job to be completed.

There are two important intuitions and facts regarding to the optimal strategy. They are also the main supports for Theorem 3

Intuition and Key Fact 1 *The selection probabilities start small and grow quickly. The purpose of the slow start is to reduce the possibility of misleading by adversary for the false start. The objective of fast growth is to ensure that we make some selections before the end of the game with high probability.*

Intuition and Key Fact 2 *The growth must be slow enough to ensure that whichever option is selected, the posteriori probability is high such that the option was selected at least P payments before the final payment.*

Theorem 3 *If P_{opt} is known and $P_{opt} > 3P \log n/q$, the strategy will return a profit $\geq P$ with probability $> 1 - q$*

Sketch of Proof We will only outline the proof in the class. The proof was based on the two intuitions and following the procedures given in [1].

Let S denote a sample space of coin flips where there will be one coin flip for each pair (x, y) for which O_x has its dividend at day y. If O_x has its i'th dividend at day y, the probability of getting head is $2^{(qi/P)}/Pn^2$. In particular, the player will schedule his job on O_x , at step y if the (x, y) coin is head and if the (x', y') coin is tail for all (x', y') such that $x' < x$ or $y' = y$ and $y' < y$. Let S_{win} denote the subspace of S consisting of sample points for which the player gets its targeted profit P or higher. In order to show the $Pr(S_{win})$, a second space S' is considered for which it is easy to show that $Pr(S')$ is large and that $Pr(S_{win})$ is nearly as large as $Pr(S')$. S' is defined to comprise of the sample points for which there is at least one Head and for which the first d flips for each O_i resulted in Tails. There are two steps involved.

1. Prove $Pr(S') \geq (1 - 1/n)$
   Because of the slow start and the fast growth, this step is to show that the probability of misleading by adversary at the beginning and the probability of not making any choice at the end are small. The probability of making a selection among the Pn flips for which $i \leq P$ is at most

$$Pn(\frac{2^q}{pn^2}) = O(\frac{1}{n})$$

The probability of making no selection at the end of the game is at most

$$
\begin{aligned}
Pr[no\ selection] &\leq \prod_{i=1}^{P_{opt}} [1 - (\frac{2^{qi/P}}{Pn_2})] \\
&< [1 - (\frac{2^{q(P_{opt}-P)/P}}{Pn_2})]^P \\
&< [1 - (\frac{n2^{-q}}{P})] \\
&< e^{-\frac{n}{2}}
\end{aligned}
$$

So $Pr(S') \geq 1 - O(1/n)$

2. Prove $Pr(S_{win}) \geq (1 - O(P \log n/P_{opt}))Pr(S')$
   It is the step to prove that S_{win} is nearly as large as S'. It is the proof step from Intuition 2 and shows that the probability of making selection at least P payments before the final payment is almost as likely as making a selection before the final payment.

   The proof of this step is to construct an injection from S' to S_{win}. Please refer to lemma 2.2 in [1] for the details.

Combining the results of step 1 and step 2, we are able to prove the theorem. If P_{opt} is known, choose target $P = qP_{opt}/(3 \log n)$ and confidence parameter $1 - q$ as desired, profit P will be obtained with probability $1 - q$. □

Theorem 3 shows the performance guarantee when P_{opt} is known. When P_{opt} is unknown, we need to make a guess of it and choose target $P = q2^R/(3 \log n)$ where R is uniformly selected from $[1, \log d]$. 2^R will approximate P_{opt} value among $2, 2^2 \ldots d$ with probability $1/\log d$. So profit $P_{opt}/(c \log n)$ will be obtained with probability $1/(c \log d)$ where c is a fixed constant. Combining the result with that in Theorem 3, we know the expected return is $cP_{opt}/(\log n \log d)$ without the knowledge of P_{opt} .

From the theorem and analysis in the section, we know that the past can be combined at random to provide a very good prediction of future performance even if the past does not correlate with the future.

3 Pick-a-Winner Game - with Variations

Now let us look at a variation of the previous game. The game has the following conditions and rules.

- K players are playing the game.
- Each player is allowed to select one option.
- There are N options that the players can select.
- There are D days that a game can be played.
- A particular dividend of an option is paid only to the highest priority player holding the particular option.
- If a player needs to change the option ,it will cost 1 $ per change.

let P_i denote the profit of the i^{th} ranked player.
Let us look at an example with 3 players and consider the following observations.

- Player *1* selects D on day *2* , swaps to E on day *5*.
- Player *2* selects D on day *1*, swaps to C on day *3*.
- Player 3 selects E on day *2*. No swaps.

The outcome would be as follows:

Results: P1 = $4, P2 = $0, P3 = $0.

	1	2	3	4	5	6	7	8	9	10
A	$1	$1								$1
B	$1		$1		$1	$1		$1	$1	
C	$1			$1						
D		$1	$1	$1	$1					
E	$1					$1				$1
F	$1					$1		$1		
G					$1					
H										

Figure 1: Game Results

After playing the game, now let us try to answer the following questions.

Q.Can Swapping Help ?
Our own intuitive answer would be **NO**. Since the the future outcome cannot be predicted , the future of one option seems to be indistinguishable from the future of another.
But further analysis shows that the answer is **YES**.The basic idea is that once the past is

seen , the randomize strategy may dictate change.
The advantages obtained by swapping are :

- Increased profits gained , where the increase in the new income surpassed the swapping cost,
- Increased probability of success,
- Knowledge of P_{OPT} is no longer needed.

Key Idea
One of the key ideas that could be gained is:
Break the time into epochs , and run independent trials in each epochs. It could be observed that with a high probability, most trials would be successful.

Q. Is Coordination needed among players to maximize one's profits?
The answer is **YES**. But very little coordination is needed.
The following items need to be considered:

- As long as no player is allowed to be too greedy, and each player can select in isolation,
- Other players are considered to be part of the adversary,
- At worst, the j^{th} ranked player can choose the j^{th} ranked option.

4 Results in Summary

In Figure 2 , summarized results of the single and multiple player Pick-a-Winner games are given.

# Players	# Picks	# Swaps	P_{OPT} known ?	Profit	Prob[success]
1	1	0	NO	C P_{OPT}/log n	1 / log d
1	1	0	YES	C P_{OPT}/log n	.99
1	1	C log n log d	NO	C P_{OPT}	1 - o(1/ n)
1	1	C log n	YES	C P_{OPT}	1 - o(1/ n)
1	O(1)	C log n	YES	(1 − o(1)) P_{OPT}	1 - o(1/ n)
K*	O(1)	C log n log P_{OPT}	NO	(1 − o(1)) P_{OPT}	1 - o(1/ n)

Figure 2: Game Results in summary

5 Applications

In this section we will be looking at some real life examples where the strategy of Pick-a-Winner game can be applied.

5.1 Scheduling a single job on NOWs

The strategy applied in single player Pick-a-Winner game can be applied in solving this problem. The problem formulation is as follows:

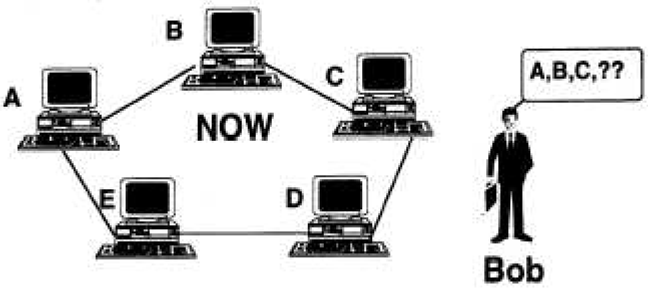

Figure 3: which workstation to select?

- Bob has a job requiring P cycles that he wants to run overnight,
- Bob picks a workstation for his job and assigns the job a background priority,
- The job gets done if the chosen workstation has at least P available cycles during the night.

The Question to answer is : How should Bob choose a Workstation?
Solution:
We will try to model the answer from the single player Pick-a-Winner game with the following mapping.

- Workstations $\rightarrow$ Options
- Cycles $\rightarrow$ Days (overnight - d days)
- Availability $\rightarrow$ Dividends paid
- Target profit $\rightarrow$ P cycles

The goal is to maximize probability of success, ***1 - q***

	1	2	3	4	5	6	7	8	9	10
A	A	A	A	A			A	A	A	A
B	A									
C					A					
D	A									
E			A							
F			A	A	A	A	A			
G					A	A				
H										

Figure 4: Availability of Workstations

In the figure 3, A denotes availability.
Results:
If some work stations will be available for at least ***3 P*** log ***n*** steps, Bob can get ***c*** log ***n*** jobs of duration ***P*** done with a probability of ***1 - o(1/n)***.
The following pointers should be considered:

- At most 1 work station is used by Bob at any time,
- Bob can kill a job in progress and restart the job elsewhere,
- The high probability of success holds for all patterns of workstations availability,
- If Bob can use **O(1)** workstations at a time, then he can attain 99 % efficiency.

5.2 Scheduling multiple Jobs on NOWs

Now we will look at scheduling multiple jobs. If there are ***K*** people , each with background jobs to run ,we can use the strategy adopted in ***K-player Pick-a-Winner*** game.

Coordination:
Each person receives a priority , but otherwise works independently.Competing demands for service would also be resolved by priority.
Now Let us look at the results.
Results:
If the j^{th} best workstation will be available for at least T_j steps, then the j^{th} ranked person will get c.T_j cycles with a high probability.
Let us look at another game.

5.2.1 Application: Video Servers

Let's formulate the problem the following way.

- During the day, d customers request movies.
- At most k out of n movies will be shown at a pre-arranged time in each evening.
- Each request is immediately accepted or rejected.
- If a request is accepted, the movie must be shown. Rejected customers are lost.

Goal:
The goal is to maximize the accepted requests thus maximizing profits.
Before looking at the game based solution, let's look at a previous solution proposed by Aggarawal, Garay and Herzberg[2].
According to their solution:

- In the worst case, $\boldsymbol{k}$ requests would be accepted .
- $\boldsymbol{c}\ P_{OPT}\ /\ \log\ \boldsymbol{n}$ requests would be accepted with some lookahead.

Now let's look at the solution that was obtained from the K-player Pick-a-Winner game. Following are the mappings.

- Movies $\rightarrow$ Options
- Customers $\rightarrow$ Days
- Requests $\rightarrow$ Dividends Paid
- Target Profit $\rightarrow$ Number of Accepted Requests.

Strategy: The strategy is to guess the number of requests R_j for j^{th} most popular movie and set target profit for j^{th} movie to be $\boldsymbol{c}\ R_j /\ \log\ \boldsymbol{n}$.
Results: The expected number of accepted requests would be =
$\boldsymbol{c}.P_{OPT}\ /\ (\ \log\ \boldsymbol{n}\ /\ \log\ \boldsymbol{d}\ )$
The following diagram denotes the requests and movies in the same analogy of playing the K-player game.

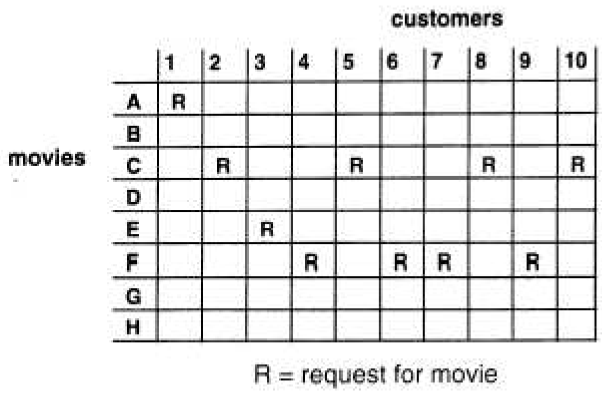

Figure 5: Movies Vs. Requests

5.2.2 Application: On-line Set Cover

Problem:
Let's formulate the problem in the following manner.

- Given $\boldsymbol{n}$ Sets $S_1, S_2 ..., S_n$.
- $\boldsymbol{d}$ items $V_1, V_2, ..., V_d$ arrive one per each step.
- As each item arrives , we are able to distinguish the set(s) the item belong to.

Goal:
The goal is to pick **k** sets to cover as many items as possible. The rewarding mechanism is as follows.

- Credit is given only once for each item, even if it is contained in several sets.
- Credit for an item is given only if the set was selected before or during the step when the item arrives.

Let's look at an example where this strategy can be applied.

Example:
A problem associated with Video server contention and availability, where each customer specifies a collection of movies , any one of which is fine.

Solution:
The solution is the same as for Video servers except that each customer can specify several movies.
The following mapping maps the problem entities to the set cover problem.

- Items → Customers
- Sets → Movies.

The following results could be obtained.
Results:
Total expected credit would be

$$cP_{OPT}/(\log n \log d)$$

The following diagram denotes the representation of the problem where the request for each movie is denoted by a set S_1 to S_n and customers are represented through the numbering from V_1 to V_d.

In Figure 6, $\in$ denotes each customer's request for a particular movie.

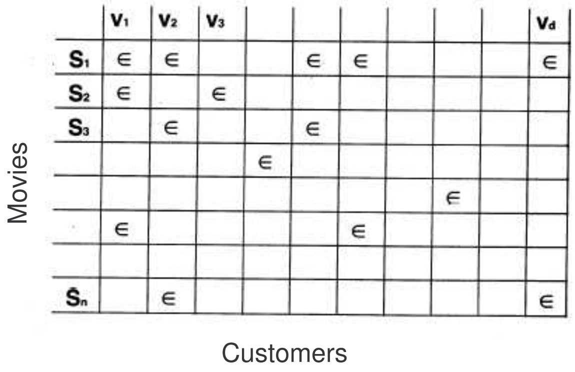

Figure 6: Movies Vs. Requests

References

[1] Baruch Awerbuch and Yossi Azar and Amos Fiat and Tom Leighton, "Making commitments in the face of uncertainty: how to pick a winner almost every time (extended abstract)", ACM, 1996

[2] Juan A. Garay, Amir Herzberg, Sudhanshu Aggarwal, "Sharing Video on Demand ", IBM,PODC, 1994