Resource Allocation in the Face of Uncertainty:

How to Pick a Winner Almost Every Time

Tom Leighton
M. I. T.

Joint work with:

Baruch Awerbuch	JHU & MIT
Yossi Azar	Tel Aviv U.
Amos Fiat	Tel Aviv U.

Talk Outline

1. The Pick-a-Winner Game

2. Strategies to maximize profit

3. Multiple players and other variations

4. Applications
   - task scheduling on networks of workstations
   - video-on-demand scheduling
   - investment planning
   - strategic planning

5. Competitive Analysis -- A broader perspective

6. Open questions

The Pick-a-Winner Game

- 1 player
- n = 8 options (A-H) to choose from
- 1 selection made at any time
- no changes allowed
- d = 10 days
- profit P = dividends paid by option <u>after</u> it was selected

Example: if D is chosen on day 4, then P = $2

Note: $P_{OPT} = 6$ (by choosing B on day 1).

	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										

The Pick-a-Winner Game (cont.)

On-Line Version

- past dividend payments are known
- future payment schedule is not known
- the past and future may not be correlated
- the future is not necessarily random -- it is determined by an adversary who knows the player's strategy

Q: How much money can the player make?

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

Simple Observations

<u>Fact:</u> The profit for any deterministic player is 0, even if P_{OPT} = \$d.

<u>Proof</u>: The adversary pays a dividend for every option on every day until the player makes a selection --- then dividends are stopped for the selected option.

<u>Remember:</u>

- We are looking at the worst-case scenario.
- The adversary knows the player's strategy.

Simple Observations (cont.)

<u>Fact:</u> We can get expected profit P_{OPT}/n by choosing a random option at the start.

<u>Proof</u>: We have a 1/n chance of selecting the optimal option.

<u>Note</u>: The adversary does not know our random numbers, only our strategy.

<u>Conjecture</u>: The random-choice-at-the-start strategy is optimal.

- Since we don't know the future, a random choice is the best that we can do.
- The past has no bearing on the future, so it can't help to delay the selection.

A Not-so-Simple Observation

The player can do much better with the right randomized strategy!

<u>Theorem</u>: The player can get profit $cP_{OPT}/\log n$ with probability .99 if he knows P_{OPT}.

<u>Remarks:</u>

- Without knowledge of P_{OPT}, the expected return is $c\, P_{OPT} / (\log n \log d)$.
- c is a constant that depends on the desired probability of success.
- The probability of success depends only on the player's random numbers.
- The probability of success holds for <u>all</u> payment schedules.
- These bounds are best possible.

Optimal Strategy

1. Player selects target profit P and confidence level 1 - q.

2. Player flips coins to decide when and what to select as follows:

<u>For</u> t = 1 to d (days)

<u>For</u> S = 1 to n (options)

<u>If</u> a selection has not yet been made, and

<u>If</u> S pays its ith dividend on day t,

<u>Then</u> we select S with probability

$$2^{(q\, i\, /\, P)} \Big/ P\, n^2$$

<u>End</u>

<u>Result:</u> If $P_{OPT} > 3\, P \log n / q$, then the strategy will return a profit >P with probability > 1 - q.

Intuition and Key Facts

The selection probabilities $2^{(q\,i\,/\,P)} / P\,n^2$ **start small and grow quickly (but not too quickly).**

- The slow start is so that the adversary can't mislead the player with false starts -- e.g., options that pay P - 1 dividends early and then nothing later.
  - Prob [mislead player] < $P\,n\left(2^q / P\,n^2\right) = O\left(1/n\right)$

Example:

	1	2	3	...	P-1	P	P+1	...	d-1	d
A_1	\$1	\$1	\$1	...	\$1					
A_2	\$1	\$1	\$1	...	\$1					
A_3	\$1	\$1	\$1	...	\$1					
.	\$1	\$1	\$1	...	\$1					
.	\$1	\$1	\$1	...	\$1					
.	\$1	\$1	\$1	...	\$1					
A_{n-1}	\$1	\$1	\$1	...	\$1					
A_n						\$1	\$1	...	\$1	\$1

Intuition and Key Facts (cont.)

We need <u>fast growth</u> to insure that we make some selection with high probability.

Prob [no selection]

$$\leq \prod_{i=1}^{P_{OPT}} \left[1 - \left(2^{(q\,i\,/\,P)} \Big/ P\,n^2 \right) \right]$$

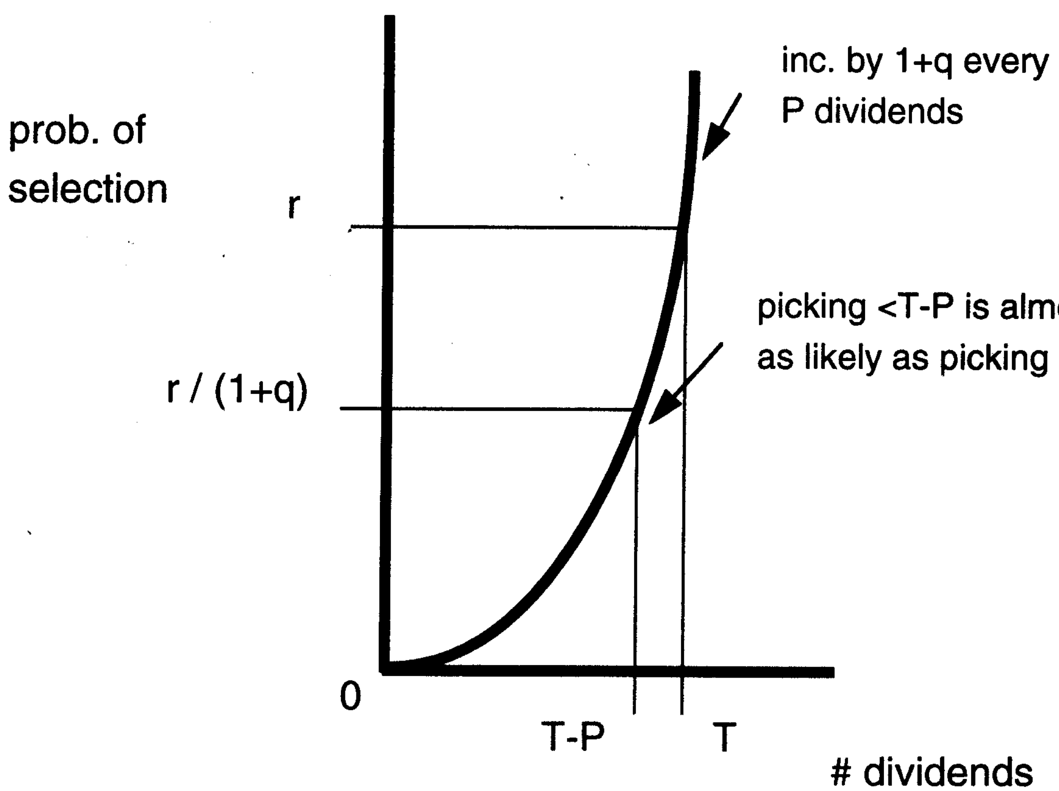
$$< \left[1 - \left(2^{(q\,(P_{OPT} - P)\,/\,P)} \Big/ P\,n^2 \right) \right]^P$$

$$< \left[1 - \left(n\,2^{-q} \big/ P \right) \right]^P$$

$$< e^{-n/2}$$

Intuition and Key Facts (cont.)

But, the growth must be <u>slow enough</u> to insure that whatever option is selected, the a postieri probability is high that the option was selected at least P payments before the final payment.



Summary

- Combining the preceding 3 conditions yields a proof...

- If P_{OPT} is known, choose target $P = q\, P_{OPT} / (3 \log n)$ and confidence parameter $1 - q$ as desired.
  - Profit P will be obtained with probability $1 - q$.

- If P_{OPT} is not known, guess it. I.e., choose target $P = q\, 2^R / (3 \log n)$ where R is uniformly selected from $[1, \log d]$.
  - 2^R will approximate P_{OPT} with probability $1 / \log d$.
  - Profit $P_{OPT} / (c \log n)$ will be obtained with probability $1 / (c \log d)$.
  - c is a fixed constant.

Summary (cont.)

- Even if the past is not correlated with the future, the past can be combined with randomness to provide a very good prediction of future performance.

- The intuition that past winners are likely to be future winners can be used as the basis for a succesful strategy, even though the intuition is not precisely correct.

The Pick-a-Winner Game -- with Variations

New Rules:

- k players
- 1 selection each
- n options
- dividends paid only to highest-priority player holding option
- d days
- changes allowed -- at cost of $1 each
- P_i = profit of ith ranked player

The Pick-a-Winner Game -- with Variations (cont.)

Example:

- 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.

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										

The Pick-a-Winner Game -- with Variations (cont.)

Let's Play!

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

Key Questions

Q: Can swapping help?

- **Intuition says No:** the future of one option seems to be indistinguishable from the future of another.
- **Analysis says Yes:** once we see the past, the randomized strategy may dictate a change.
- **Advantages obtained by swapping:**
  - Increased profit
  - Increased probability of success
  - Knowledge of P_{OPT} no longer needed

Key Idea: break time into epochs, and run independent trials in each epoch.

- With very high probability, most trials will be successful.

Key Questions (cont.)

Q: Is coordination needed among players?

A: Yes, but very little.

- As long as no player is allowed to be too greedy, each player can make selections in isolation.
- Other players are considered to be part of the adversary.
- At worst, the jth ranked player can choose the jth ranked option.

Summary of Results

#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	$(1 - o(1))\, P_{OPT}$	$1 - o(1 / n)$

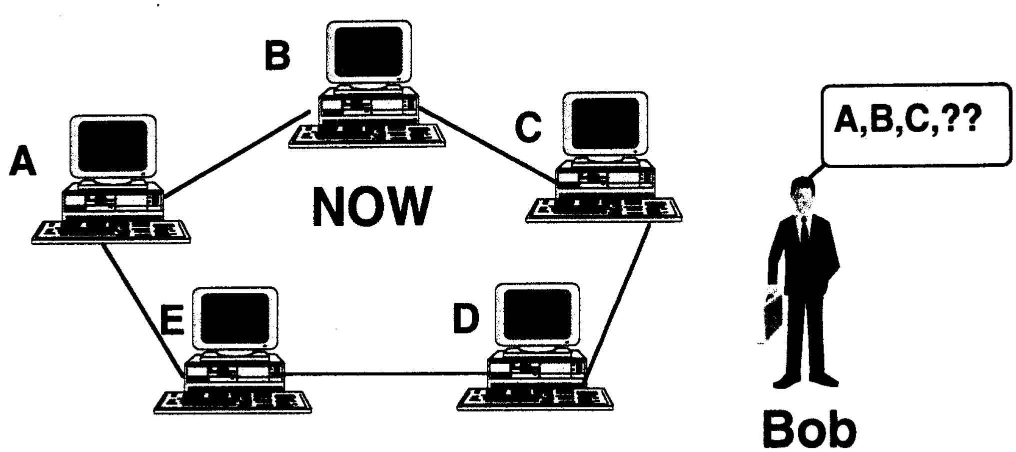
* jth ranked player gets 99% of profit of jth ranked option.

Application: Scheduling on NOWs

Example Problem:

- 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.

Q: Which workstation should Bob choose?



Scheduling on NOWs (cont.)

Solution:

Model problem as 1-player Pick-a-Winner Game

- workstations ~ options
- cycles ~ days (overnight ~ d days)
- availability ~ dividends paid
- target profit = P cycles

Goal: 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										

A denotes availability

Scheduling on NOWs (cont.)

Result:

If some workstation will be available for at least 3 P logn steps, Bob can get c logn jobs of duration P done with probability 1 - o(1 / n).

Remarks:

- At most 1 workstation 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 workstation availability.
- If Bob can use O(1) workstations at a time, then he can attain 99% efficiency.

Scheduling on NOWs (cont.)

Multiple Jobs

If there are k people, each with background jobs to run, then we can use the k-player Pick-a-Winner strategy.

Coordination:

- Each person receives a priority, but otherwise works independently.
- Competing demands for service are resolved by priority.

Result:

If the jth best workstation will be available for at least T_j steps, then the jth ranked person will get $c\ T_j$ cycles with high probability.

Application: Video Servers

Example Problem:

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

GOAL: maximize the number of accepted requests.

Prior Results:

- k accepted requests in the worst case.
- c P_{OPT} / logn accepted requests with some lookahead.

Aggarwal-Garay-Herzberg (IBM '94, PODC '94)

Video Servers (cont.)

Our Solution:

Model problem as k-player Pick-a-Winner Game

- movies ~ options
- customers ~ days
- requests ~ dividends paid
- profit ~ number of accepted requests

Strategy: guess number of requests R_j for jth most popular movie & set target profit for jth movie to be $c\ R_j / \log n$.

Result: Expected number of accepted requests $= c\ P_{OPT} / (\log n \log d)$.

customers

movies

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

R = request for movie

Application: On-Line Set Cover

Problem:

- Given n sets $S_1, S_2, \ldots, S_n$.
- d items $v_1, v_2, \ldots, v_d$ arrive one-per-step.
- As each item arrives, we learn to which set(s) it belongs.

GOAL: pick k sets to cover as many items as possible.

- 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.

Example:

- video server problem where each customer specifies a collection of movies, any one of which is OK.
- investment planning.

On-Line Set Cover (cont.)

<u>Solution</u>:

Same as for video servers except that each customer can specify several movies.

- **items ~ customers**
- **sets ~ movies**

<u>Result:</u> Total expected credit is

$c P_{OPT} / (\log n \log d)$

	V_1	V_2	V_3							V_d
S_1	∈	∈			∈	∈				∈
S_2	∈		∈							
S_3		∈			∈					
				∈						
								∈		
	∈					∈				
S_n		∈								∈

∈ denotes set membership

Application: Strategic Planning

Example Problem:

- A general wants to find a soft spot among n enemy defensive positions.
- The enemy cannot afford to simultaneously defend all n positions.
- The enemy can rearrange defensive forces daily, but movement of forces is expensive.
- The general knows where the enemy forces were each day in the past -- but not where they will be tomorrow.
- The attack must take place within d days.

GOAL: decide where and when to attack so as to be assured victory.

Difficulty: Precise timing of the attack requires a more sophisiticated algorithm.

Application: Strategic Planning (cont.)

Solution: Use a modified 1-player Pick-a-Winner Strategy with target profit 1.

Modification:

- When deciding whether or not to attack a position, use probability

$$2^{(q\,i)} / (n\,d)^2$$

where i is the number of consecutive preceding days that the position has been left undefended.

Result: The attack will be succesful with probability 1-q if the enemy leaves some site undefended for c log(nd) / q consecutive days.

On-Line Algorithms and Competitive Analysis

Goal:

Develop good algorithms for decision-making in the face of an uncertain future.

Performance Measures:

An algorithm is competitive if for any manifestation of the future, the algorithm does nearly as well as possible.

Example:

The best algorithm for the Pick-a-Winner Game is (c logn logd) - competitive.

- I.e., not knowing the future costs you a (c logn logd) factor in profitability.

Open Questions

How far can we go in predicting the future?

Is there a competitive algorithm for the stock market problem where prices can go up <u>and</u> down?

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