10-2015

The Effectiveness of Using a Modified "Beat Frequent Pick" Algorithm in the First International RoShamBo Tournament

Proceso L. Fernandez Jr
Ateneo de Manila University, pfernandez@ateneo.edu

Sony E. Valdez

Generino P. Siddayao

Recommended Citation

Sony E. Valdez, Generino P. Siddayao, and Proceso L. Fernandez, "The Effectiveness of Using a Modified "Beat Frequent Pick" Algorithm in the First International RoShamBo Tournament," International Journal of Information and Education Technology vol. 5, no. 10, pp. 740-747, 2015.

The Effectiveness of Using a Modified "Beat Frequent Pick" Algorithm in the First International RoShamBo Tournament

Sony E. Valdez, Generino P. Siddayao, and Proceso L. Fernandez

***Abstract*—In this study, a bot is developed to compete in the first International RoShamBo Tournament test suite. The basic "Beat Frequent Pick (BFP)" algorithm was taken from the supplied test suite and was improved by adding a random choice tailored fit against the opponent's distribution of picks. A training program was also developed that finds the best performing bot variant by changing the bot's behavior in terms of the timing of the recomputation of the pick distribution. Simulation results demonstrate the significantly improved performance of the proposed variant over the original BFP. This indicates the potential of using the core technique (of the proposed variant) as an Artificial Intelligence bot to similarly applicable computer games.**

***Index Terms*—Artificial intelligence, game theory, rock paper scissors, RoShamBo.**

I. Introduction

In this paper, a modified version of the "Beat Frequent Pick (BFP)" algorithm is presented that is used in the First International RoShamBo Programming Competition.

A. What Is Rock Paper Scissors?

Rock Paper Scissors is an intransitive two-player hand gesture game. The objective of the game is to defeat your opponent with a choice of hand gesture. The different hand gestures or options are "Rock, Paper, and Scissors". Each of the players select an option in secret. After both players have selected an option, they play their choice as their move for that turn. Rock Paper Scissors is a double-blind game, so both players reveal their move at the same time. The winner of the turn is then determined with the matrix in Table I.

Rock Paper Scissors is known by different names such as RoShamBo, Bato Bato Pik, Jak-en-poy, and Quartz Parchment Shears. For this paper, the name RoShamBo is used as this is the name chosen for one of the Artificial Intelligence (AI) competition available on the Internet.

B. What Is the International RoShamBo Programming Competition?

Darse Billings announced the First International RoShamBo Programming Competition on September 1999 [1] and the results were published in the International Computer Games Association (ICGA) Journal [2].

TABLE I: RPS Victory Conditions

	P2 plays rock	**P2 plays paper**	**P2 plays scissors**
P1 plays rock	Tie	P2 wins	P1 wins
P1 plays paper	P1 wins	Tie	P2 wins
P1 plays scissors	P2 wins	P1 wins	Tie

To participate in the programming competition, competitors are tasked in creating an AI bot using the C programming language and have it return 0, 1 or 2 (respectively representing rock, paper, scissors). Each bot will compete against all participating bots in a series of matches, each comprising 1000 turns. In addition, all bots have access to the history of moves played by both players during the current match-up.

So far, there are two International RoShamBo Programming Competitions: one on September 1999 and another on July 2000.

1) RoShamBo ranking system

The RoShamBo tournaments has two ranking system: The number of turns won (tournament results) and the number of matches won (match results).

Each match will play for 1000 turns.

In the tournament ranking system, the points that each AI has accumulated over the entire tournament is recorded. The more turns won, the higher the AI's ranking is.

In the match ranking system, the match points are computed by subtracting the number of turns lost from the number of turns won in the match. For example, if the player has 437 wins and 261 loses, his final points for that match is 176. For this ranking, a break-even point is first established. For the first RoShamBo Tournament, the break-even point is set at 50. This means that receiving a score between -50 and +50 will result in a tie. Thus, the AI will only receive a match win if the bot has won at least 51 points in the match, a loss for games with less than -50 points, and a tie otherwise. Winning a match gives 2 ranking points, receiving a tie gives 1 ranking point and losing a match gives 0 ranking points. The more ranking points the AI has, the higher the AI's ranking is.

2) The first RoShamBo tournament test suite

For the purpose of this paper, the first RoShamBo Tournament was chosen instead of the latest RoShamBo Tournament. The rationale in this is to avoid competition with AIs that are countering against the *Iocaine Powder* Meta [3].

Manuscript received May 4, 2014; revised July 10, 2014.

S. E. Valdez is with the Agoo Computer College, Agoo, La Union, Philippines (e-mail: shunyvaldez@gmail.com).

G. P. Siddayao is with the Cagayan State University-College of Information and Computing Sciences, Tuguegarao City, Philippines 3500 (e-mail: genersiddayao@gmail.com).

P. L. Fernandez is with the Ateneo de Manila University, Quezon City, Philippines (e-mail: pfernandez@ateneo.edu).

Due to the open source nature of the tournament, bots in later tournaments has a meta-strategy of exploiting patterns from the previous winners. To be fair, exploiting the meta is a valid strategy. But it is the objective of this research to create a generic AI bot.

Thus, to have a clearer view on the winning AI strategies, we choose to play with the first RoShamBo Tournament where a meta has not yet been priorly formed.

II. Definitions

Bots. Bot was originally an abbreviation for robot, but has changed to mean a computer software that uses Artificial Intelligence.

Option. An option refers to the set of valid choices that a player can chose from. In RoShamBo, the options for both players are Rock, Paper, and Scissors.

Move. A move is a play in one turn that has been selected by a player from amongst the possible options to select from.

Turn. A turn is a moment during a game when a player have to select a move that they believe will cause them to achieve a victory condition.

Match. An instance of a game between two or more players. A match is over when the winners and losers have been declared, as defined by the rules of the game.

Tournament. A tournament is an event where multiple players play against each other to determine their rank. A tournament rank can be determined through different tournament formats such as *Single Elimination*, *Double Elimination*, and *Round Robin*.

Meta-strategy. Meta means to "think beyond". A meta-strategy (commonly shortened to meta) is a strategy which takes into account the current strategies that are either dominating the game or are very common.

III. RoShamBo AI

Two major approaches have been used to produce strong RoShamBo players: purely statistical techniques and the direct history method [3]. In purely statistical techniques, a strategy is chosen and statistical techniques are developed to model that strategy. In direct history, the move history from both players are analyzed for patterns.

The First International RoShamBo Programming Competition test suite contains multiple AIs that includes the high ranking AIs from the first tournament. The following AIs are relevant to this paper:

A. Beat Frequent Pick

The Beat Frequent Pick (BFP) is a "dummy bot" that was created to serve as a basic example of a RoShamBo AI. This AI operates on the idea that a player will favor a specific move (for example, rock). The AI then records a running tally of the number of times rocks, papers, and scissors are used in order to determine which move the opponent favors. When selecting a move, the BFP bot simply chooses the move that beats the opponent's most used move.

B. Inocencio

In the RoShamBo tournament, the Inocencio bot is closest to the researcher's reinterpretation of the BFP algorithm. However, while the developed AI calculates the probability over the entire match, the Inocencio bot calculates the probability over a sliding window of the previous *n* turns (where *n* is 20 in the test suite).

When the Inocencio bot detects that its opponent uses a move with a probability of > 0.45, it will immediately assume that the opponent will use that move otherwise, the bot will use a random move with a bias on the opponent's probable move. The researchers' AI do not have confidence checks.

Finally, the researchers' AI tries to predict the future (1...1000 turns into the future) while the Inocencio bot does not.

C. Iocaine Powder

The Iocaine Powder bot is the AI that ranked highest in the first International RoShamBo Programming Competition.

The Iocaine Powder bot uses multiple strategies and predictive algorithm to select a move [3], [4]. When playing a match, it keeps score on which prediction wins and uses that algorithm. It has three prediction algorithms:

1) History Matching. The bot will study the last *n* moves (where *n* can be 1, 2, 5, 10, 100, or 1000) and try to look for move combinations that were played in the past.
2) Frequency Analysis. The algorithm is similar to the Beat Frequent Pick bot.
3) Random Guess. Once the bot detects that it is losing (determined by a threshold), the bot will start predicting randomly.

For strategies on how to play a turn, it has 6 strategies:

p.0. Naive application. Assume that the bot's prediction is correct and play the winning move (for example, rock was predicted so the bot uses paper).

p.1. Defeat second guessing. Assume that the opponent will counter p.0, so play the move that beats the winning move (continuing the above example, the opponent will second guess and choose paper, so the AI will choose scissor).

p.2. Defeat triple guessing. Assume that the opponent will counter p.1, so play the move that beats the winning move (continuing the above example, the opponent will triple guess and choose rock to beat p.1's scissor, so the AI will choose paper).

p'.0. Second guess the opponent's move. Assume that the opponent will choose p.0. and play against that move.

p'.1. Second guess the opponent's move. Assume that the opponent will choose p.1. and play against that move.

p'.2. Second guess the opponent's move. Assume that the opponent will choose p.2. and play against that move.

D. Meta-Strategy

Due to the nature of competition, players in competitive games will develop meta-strategies.

For example, in the first RoShamBo tournament, the Iocaine Powder was the winning bot. This had direct influence on new entries in the second RoShamBo. Many of the entries in the second contest were modeled from Iocaine Powder [3]. As such, the meta for the second tournament was to beat the Iocaine Powder AI.

When playing to win in a competition, it is necessary for a player to exploit the current meta. However, this can

introduce additional complexities to the starting development of a bot: should a researcher develop a bot that aims to win but requires studying the meta, or should a researcher develop the bot in isolation from the meta and then add improvements as dictated by the meta? We have chosen the latter strategy.

E. Research Questions

1) Can we develop an AI that can perform well in the first International RoShamBo Tournament test suite?
2) How do we inject new behavior into the AI?
3) Amongst the different AI variants, what is the best performing variant of the bot?

IV. Related Literature

A. AI in General

Artificial Intelligence is a branch of computer science where machines and software are developed with human-like intelligence. The field of AI was founded at the Dartmouth College conference and was first coined by John McCarthy [5].

Artificial Intelligence has been used in numerous fields and commercial products such as speech recognition [6], natural language processing [7], and computer games [8], [9].

An AI can follow a predefined set of rules. However, this will make the bot deterministic and predictable [10]. A human player is unpredictable and an AI should also be as well. To make AI unpredictable, random numbers are used. For example, Go AI uses Monte Carlo to influence its problem analysis [11].

When encountering a problem, a human will first analyze the problem. These analysis are taken into consideration when making a decision [12]. An AI can mimic this model by implementing a problem analysis component and a decision-making component

B. AI in Games

Emergent Behavior can occur from simple rules. The AI in the video game, The Sims, follows simple rules. From these rules, the Sims behaved in a way that was not pre-programmed by the AI developer [13].

AI was also developed for the video game, Cut the Rope. Here, AI can create a level using a simulation based approach [14]. An AI was created that can learn to play Atari Games [15]. A group of bots that exhibit human behavior has been programmed into the Quake 3 video game [10].

Traditional games such as Chess and Poker also have AIs developed by numerous researchers. In Chess, IBM's Deep Blue defeated then World Chess Champion Garry Kasparov [16]. In Poker, a testbed was created to aid machine intelligence research [17].

C. Publications in RoShamBo

Research has found that humans imitate opponent's gestures as a strategy [18]. Mathematics has developed a model that learns the game [19]. Robots are created that plays RoShamBo [20].

Zhijian Wang observed that winning players stick to their one winning strategy while losing players changes strategies [21]. Psychology has conceptualize the "conditional response" [22] and game theory has the "Pavlov strategy" [23].

V. Methodology

A. Research Methodology

1) Create an Artificial Intelligence RoShamBo bot based on a strategy.
2) Create a training program for the AI bot. This training program should modify the behavior of the AI bot.
3) Each modified variant will play in the first International RoShamBo Programming test suite[1]. The rankings are recorded and the process is repeated. After a predetermined limit is reached, the best performing version of the bot is determined.

B. Modified Beat Frequent Pick AI (MBFP)

We based our AI on the idea that the best strategy for winning RoShamBo is to keep a uniform spread between rock, paper, and scissors [19]. Meaning, in a 9-turn game, there will be 3 rocks, 3 papers, and 3 scissors. The AI on this paper will model our opponent's moves as if they are using this strategy and in addition, will try to predict the future by computing the probability every *n* turns.

During the match, the AI will keep track of the number of rocks, papers, and scissors the opponent has played. We will call these $stat_R$, $stat_P$, and $stat_S$, respectively. We also keep track of the current turn with the variable *currentTurn*.

For the initial turn, the probability for all moves are set to 1/3.

When selecting a move, the AI will predict what the opponent will use. It will randomly choose between rock, paper, and scissors, with a bias on moves the opponent favors. For example, if the opponent uses rock all the time, $prob_R$ will have a value of 1.0 while the other variables will have a value of 0.0.

To predict future plays from the opponent, the AI will compute the probability every *n* turns. In effect, there is a sliding window on when the probability is recomputed. We will call these *n* values, the *targetPredictionSize*. This would allow the AI to look ahead into the future based on its current model of the opponent's probability options. We have identified that *targetPredictionSize* determines the behavior of the AI and as such, can be trained with an external training program.

Two *prediction variables* are used to give the AI a future prediction on the probability for rocks, papers, and scissors. These variables will be called $prediction_R$ and $prediction_P$. Because RoShamBo only have 3 different moves, $prediction_S$ can be inferred.

The *prediction variables* are recomputed if any the following conditions are satisfied:

1) Every *n* turn (where *n* is the AI's *targetPredictionSize*);

[1]One AI in the test suite was changed for the purpose of this research: *Shofar*. *Shofar* has a one-off bug which causes it to throw an exception (line 3021 in the unmodified source code). The researchers sidestep the problem by modifying the assert, but we note that the random numbers that causes the exception can sometimes return an invalid move. The test suite, however, will modulo any invalid move as to not interfere with the tournament.

This condition will change the AI's timing on when to re-evaluate its predictions or

2) If the *prediction variables* this turn are outside the [0..1] range; This can happen if one of the target probability is below the actual stat (for example, $stat_R$ has a value of 151 but $prediction_R$ has a value of 150.19). This indicates that the current prediction model is complete and new *prediction variables* needs to be calculated.

Because of the nature of the algorithm, if *targetPredictionSize* = 1 ($MBFP_1$), the AI will not look into the future and will instead predict the opponent based on their current and previous plays.

Finally, the AI will play the move that will beat the prediction (e.g. the AI predicts the opponent will play rock so it plays paper).

The program's flowchart and code listing can be found at Fig. 1 and Fig. 2, respectively.

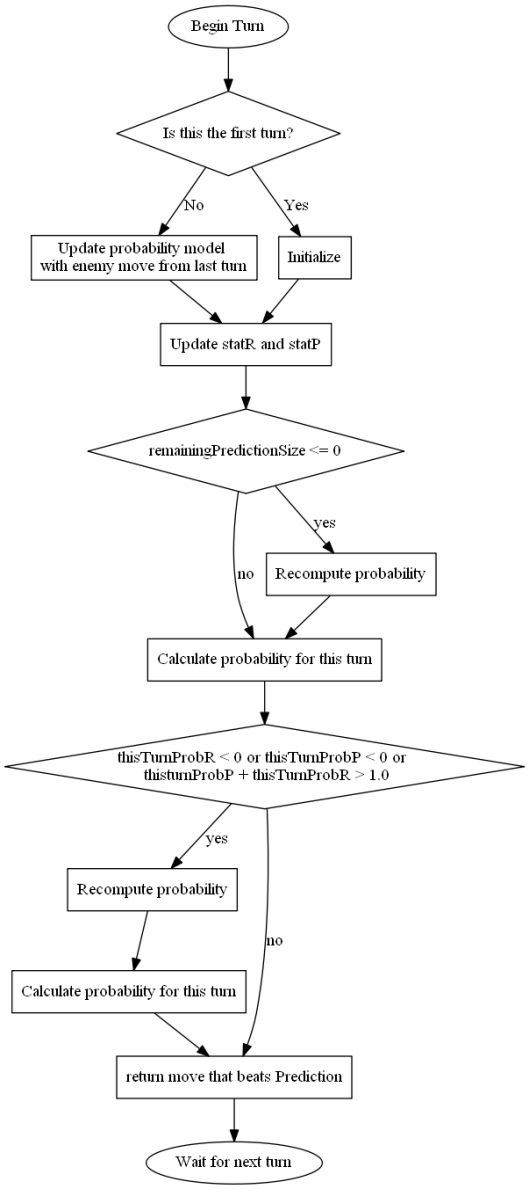

Fig. 1. Program flowchart.

C. The Training Program

We exposed the *targetPredictionSize* variable to external programs by adding and modifying the code in the original test suite. This enables the test suite to check the program's arguments and pass the values to the bot.

AI() function
/ currentTurn refers to the how many turns has passed */*
if currentTurn == 0:
  call the *Initialize subroutine*
targetPredictionSize = [1..1000]
Prediction = *Predictor()* function
/ Play the move that beats the prediction */*
Move = (Prediction + 1) % 3
return Move
Initialize subroutine
$stat_R$, $stat_P$ = 0.0, 0.0
$prob_R$, $prob_P$ = 1/3, 1/3
$prediction_R$ = $prob_R$ X targetPredictionSize
$prediction_P$ = $prob_P$ X targetPredictionSize
remainingPredictionSize = targetPredictionSize
Predictor() function
if opponent used Rock last turn: $stat_R$ += 1
if opponent used Paper last turn: $stat_P$ += 1
if remainingPredictionSize <= 0:
  call the *RecomputeFutureProb subroutine*
call the *CalculateProbThisTurn subroutine*
if $thisTurnProb_R$ < 0.0 or $thisTurnProb_P$ < 0.0 or
  $thisTurnProb_R$ + $thisTurnProb_P$ > 1.0:
{
  call the *RecomputeFutureProb subroutine*
  call the *CalculateProbThisTurn subroutine*
}
remainingPredictionSize -= 1
return *biased_roshambo function*($thisTurnProb_R$, $thisTurnProb_P$)
RecomputeFutureProb subroutine
$prob_R$ = $stat_R$ / currentTurn
$prob_P$ = $stat_P$ / currentTurn
$prediction_R$ = $prob_R$ X (currentTurn + targetPredictionSize)
$prediction_P$ = $prob_P$ X (currentTurn + targetPredictionSize)
remainingPredictionSize = targetPredictionSize
CalculateProbThisTurn subroutine
$thisTurnProb_R$ =
  ($prediction_R$ – $stat_R$) / remainingPredictionSize
$thisTurnProb_P$ =
  ($prediction_P$ - $stat_P$) / remainingPredictionSize
biased_roshambo($prob_R$, $prob_P$) function
throw = random() / MAXRANDOM
if throw < $prob_R$:
  return 0 */* Rock */*
else if throw < $prob_R$ + $prob_P$:
  return 1 */* Paper */*
else:
  return 2 */* Scissors */*

Fig. 2. AI code listing.

Because the set of possible *targetPredictionSize* is limited (from 1 to *maxturns*), we used a linear value function to decide on the values instead of more sophisticated approach such as genetic algorithms or reinforcement learning. Our training program was developed in Python and when executed, it replays the tournament using different *targetPredictionSize*. The tournment result data for each AI variant is sent to a database.

Another Python script will parse the database and extract the ranking of each AI variant which is then rendered into a chart.

As shown in Fig. 3, all of the above scripts are what the training program is made of.

D. Findings

Against the 41 AIs from the First International RoShamBo Test Suite, the best performing MBFP variants has a

respective tournament ranking and match ranking of 23 (variant $MBFP_1$) and 21 (variant $MBFP_2$).

For variants with a *predictionSizes* of 15 to *maxturns*, we see a decline in ranking.

This indicates that the MBFP strategy is more effective if it immediately assessed the current situation instead of assessing future situations.

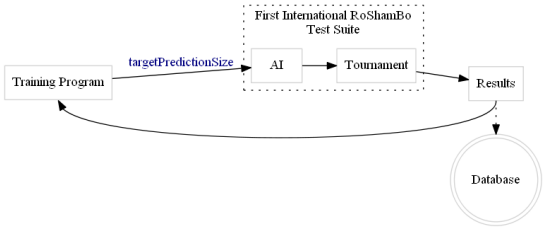

Fig. 3. Training flowchart.

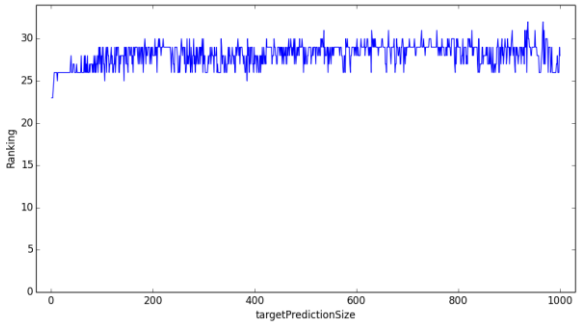

Fig. 4. Tournament results (lower is better).

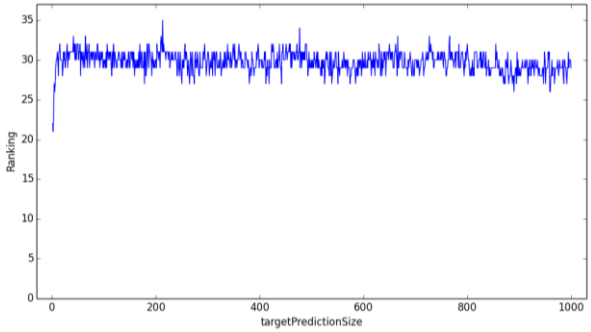

Fig. 5. Match results (lower is better).

TABLE II: WIN-LOST-TIE RECORDS ON SELECTED AIS

AI Opponent	Wins	Losts	Ties
Good Ole Rock	1000	0	0
R-P-S 20-20-60	950	0	50
Rotate R-P-S	13	0	987
Beat Frequent Pick	4	884	112
Iocaine Powder	9	167	824
Inocencio	0	965	35

TABLE III: SELECTED MBFP ONE-VS-ONE PERFORMANCES (BEST VARIANTS)

AI Opponent	Best *targetPrediction*	Best Score
Good Ole Rock	$MBFP_1$	100
R-P-S 20-20-60	$MBFP_{66}$	187
Rotate R-P-S	$MBFP_3$	341
Beat Frequent Pick	$MBFP_2$	69
Iocaine Powder	$MBFP_{575}$	84
Inocencio	$MBFP_2$	32

TABLE IV: SELECTED MBFP ONE-VS-ONE PERFORMANCES (WORST VARIANTS)

AI Opponent	Worst *targetPrediction*	Worst score
Good Ole Rock	$MBFP_{999}$	596
R-P-S 20-20-60	$MBFP_{972}$	34
Rotate R-P-S	$MBFP_{516}$	-40
Beat Frequent Pick	$MBFP_{628}$	-286
Iocaine Powder	$MBFP_{169}$	-250
Inocencio	$MBFP_{573}$	-433

Fig. 4 and Fig. 5 show the position of each MBFP variants in the tournament ranking system and the match ranking system, respectively.

E. Insights and Observations

During the course of the research, we have gained insights on our AI's behavior against other AIs. We have written up these observations on these match-ups. Table II contains the win, lose, and tie records against these opponents. Table III and Table IV contains the best and worst MBFP variant amongst the specific opponents, respectively.

1) One-vs-one against other AIs

When fighting the *Good Ole Rock* (a dummy bot included in the test suite that only plays Rock), the BFP bot has a high rate of winning, even with different *targetPredictionSize*. Because of the simplicity of *Good Ole Rock*, the BFP bot was able to quickly model the opposing AI's strategy. Against this bot, our AI has won with all of its 1000 variants.

When fighting bots that play using probability, our AI has a good chance of modeling and defeating these bots. This can be checked by looking at the score against an AI that uses this strategy such as the *R-P-S 20-20-60* bot, where our AI has won 950 matches, no losts, and 50 ties.

The test suite also contains bots that play using patterns. One example is the *Rotate R-P-S* which cycles between rock, paper, and scissors. Against this opponent, our MBFP bot tied in the majority of the turns. Our AI has 13 winning variants, no losts, and 987 tied variants. Our bot is poor at detecting patterns. If the researchers want to expand their bot to predict patterns, they would need to explore the direct history method that has become common in the second International RoShamBo Tournament [3].

Also included in the test suite are bots that uses statistical techniques. Our MBFP AI uses a statistical techniques as well. When fighting against our inspiration, the *Beat Frequent Pick*, our AI has 4 variants that won, 884 variants that lost, and 112 variants that tied. This indicates that our bot is predictable as it was successfully modeled by the basic BFP implementation.

Against the highest ranking bot, the *Iocaine Powder* AI, our AI has receive many losts. The best score against the *Iocaine Powder* is 84. The worst score against the *Iocaine Powder* is -250. This means that while more work is needed to defeat the leading AI, it is achievable.

Against our bot's closest implementation, *Inocencio*, our AI has 0 wins, 965 variants that lost, and 35 variants that tied. This adds credence to the idea that our bot is susceptible to prediction. An alternative explanation for the losts can be that *Inocencio*'s strategy of modeling the past using a sliding

window is superior to our strategy of predicting the future by modeling probability. A future study can be done on increasing the *Inocencio* sliding window similar to how our bot's *targetPredictionSize* is modified by the training program. Interestingly, *Inocencio* loses to *R-P-S 20-20-60* with a score of -322, while our bot did not received a lost. We theorized that perhaps adding a sliding window that checks the past, similar to Inocencio, will improve our AI's score.

2) BFP tournament performance insights

An AI consists of a problem analysis system and a decision making system. MBFP is a problem analysis system. Our AI uses a simple decision making system: take the prediction from the MBFP and return the move that beats it. But even with a basic decision making algorithm, the MBFP still has a decent win rate.

The champion bot, *Iocaine Powder* has a more sophisticated decision making component. This causes it to beat our AI, with the best variant using $MBFP_{575}$. This shows that developing a more robust decision making system will be a good focus for improving the AI.

In terms of ranking, $MBFP_1$ and $MBFP_2$ are the best performing bot variant when ranking for tournament and match results, respectively.

We have noticed that in the first International RoShamBo tournament test suite, there are no bots that attempts to trick, bait, and trap their opponent. Meaning, our AI has not encountered bots that intentionally play moves to skew the probability model. The researchers believe that if such an AI is encountered, the above variants may not perform as well as other values. The researchers limited themselves to the first test suite to reduce complexity but this train of thought can be explored in future papers.

A complete list of the best and worst MBFP variants and how they performed can be found at Table V.

3) Other observations

The researchers have found it interesting that just predicting the opponent's move history is enough for the AI to perform well. The other AIs, in addition to modeling their opponent's history, also model their opponent's possible winning strategies. We suspect that the MBFP's effectiveness is due to the symmetric gameplay of RoShamBo, meaning that both players have access to the same moves. As such, one strategy can work with either players. We theorized that in asymmetrical games, modeling the opponent's possible winning strategies will have a more pronounced effect on an AI's performance.

The International RoShamBo tournament is not meant for research purposes. Its marketed more as a programming competition. As such, most of the AIs in the test suite are not fully explained, and uses code optimization techniques (which makes it hard to read code in some cases). This makes it difficult to dive deep into the code of some AIs. As such, our understanding of how some of the opposing AI work is incomplete. However, this does not invalidate the result of this paper as the researchers are using the ranking system as the basis of the AI strength. If the researchers were to create a bot that aims to beat the tournament meta, more time will be spent dissecting and de-obfuscating the opposing bots.

F. Source Code

TABLE V: MBFP ONE-VS-ONE PERFORMANCES (COMPLETE RESULTS)

AI Opponent	Best *target Prediction*	Best Score	Worst *target Prediction*	Worst Score
Good Ole Rock	$MBFP_1$	1000	$MBFP_{999}$	596
R-P-S 20-20-60	$MBFP_{66}$	187	$MBFP_{972}$	34
Rotate R-P-S	$MBFP_3$	341	$MBFP_{516}$	-40
Beat The Last Move	$MBFP_{858}$	42	$MBFP_{217}$	-90
Always Switchin	$MBFP_3$	107	$MBFP_{235}$	-34
Beat Frequent Pick	$MBFP_2$	69	$MBFP_{628}$	-286
Pi	$MBFP_{12}$	25	$MBFP_{96}$	-41
Switch A Lot	$MBFP_3$	64	$MBFP_{472}$	-51
Flat	$MBFP_3$	170	$MBFP_{299}$	-31
Anti-Flat	$MBFP_1$	993	$MBFP_{211}$	-108
Foxtrot	$MBFP_5$	63	$MBFP_{83}$	-15
De Bruijin	$MBFP_{59}$	44	$MBFP_{182}$	-52
Text	$MBFP_{92}$	74	$MBFP_{203}$	-45
Anti-rotn	$MBFP_{326}$	71	$MBFP_{15}$	-98
Copy-drift	$MBFP_{197}$	100	$MBFP_{193}$	-72
Add-react	$MBFP_{30}$	69	$MBFP_{988}$	-85
Add-drift	$MBFP_{330}$	53	$MBFP_{43}$	-63
Iocaine Powder	$MBFP_{575}$	84	$MBFP_{169}$	-250
Phasenbott	$MBFP_{284}$	98	$MBFP_{363}$	-141
MegaHAL	$MBFP_1$	39	$MBFP_{616}$	-409
RussRocker4	$MBFP_{869}$	85	$MBFP_{39}$	-178
Biopic	$MBFP_{98}$	58	$MBFP_{286}$	-375
Simple Modeller	$MBFP_2$	7	$MBFP_{645}$	-491
Simple Predictor	$MBFP_1$	-12	$MBFP_{566}$	-427
Robertot	$MBFP_2$	-13	$MBFP_{620}$	-422
Boom	$MBFP_{522}$	45	$MBFP_{680}$	-307
Shofar	$MBFP_1$	11	$MBFP_{309}$	-303
ACT-R Lag2	$MBFP_{880}$	49	$MBFP_{146}$	-199
Majikthise	$MBFP_{437}$	72	$MBFP_{98}$	-79
Vroomfondel	$MBFP_{621}$	70	$MBFP_{174}$	-97
Granite	$MBFP_2$	-6	$MBFP_{897}$	-450
Marble	$MBFP_2$	54	$MBFP_{924}$	-416
ZQ Bot	$MBFP_{842}$	30	$MBFP_{370}$	-318
Sweet Rocky	$MBFP_{624}$	61	$MBFP_{314}$	-357
Piedra	$MBFP_{693}$	47	$MBFP_{900}$	-349
Mixed Strategy	$MBFP_{515}$	41	$MBFP_{960}$	-349
Multi-strategy	$MBFP_1$	10	$MBFP_{885}$	-450
Inocencio	$MBFP_2$	32	$MBFP_{573}$	-433
Pterbot	$MBFP_{33}$	69	$MBFP_{128}$	-253
Bugbrain	$MBFP_{626}$	65	$MBFP_{662}$	-127
Knucklehead	$MBFP_{207}$	43	$MBFP_{27}$	-148

The researchers have uploaded the results from our tests online, as well as tools to parse the results [24].

VI. Conclusion

In this paper, the researchers have developed an Artificial Intelligence to play in a RoShamBo tournament. They identified the variables that controls the behavior of the bot and they created a training program that finds the highest ranking AI variant when matched against the participants of the first International RoShamBo Programming Competition test suite.

A. Future Directions

Training program. In this paper, the researchers have demonstrated how changing one variable has an effect on its overall effectiveness. Future bots can be developed with a more sophisticated training program that feeds an AI different values that can effect the behavior and its ranking.

Building a new RoShamBo AI. If future researchers are to build AIs that can take part in the RoShamBo tournament, our MBFP variant should serve as the ranking to beat.

Applying the MBFP to other games. At its core, the MBFP is a bot that tries to predict the next move an opponent will play. This can be applied to other competitive games where player has to select a move that can score a point against opposing players. Examples of competitive games are Chess, Poker, Pokémon, and Street Fighter. In these games, a player can win by out-predicting and out-strategizing their opponent.

Decision making component. At its core, the MBFP is a problem analysis algorithm. It takes the database of moves used by both players and returns a prediction. The decision making algorithm is very simple: take the result from the MBFP and select a move that beats the prediction.

Acknowledgment

This work was supported in part by the Commission on Higher Education of the Philippines and the University of Cordilleras, Baguio City, Philippines.

References

[1] D. Billings. (Oct. 1999). The first international roshambo programming competition." revised July 2000. [Online]. Available: http://webdocs.cs.ualberta.ca/~darse/rsbpc1.html
[2] D. Billings, "The first international roshambo programming competition," *ICGA Journal*, vol. 23, no. 1, 2000.
[3] D. Billings. (March 20, 2001). The second international roshambo programming competition report. [Online]. Available: http://webdocs.cs.ualberta.ca/~darse/rsbpc.html#REPORT
[4] D. Egnor, "Iocaine powder," *ICGA Journal*, vol. 23, no. 1, 2000.
[5] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A proposal for the Dartmouth summer research project on artificial intelligence, august 31, 1955," *AI Magazine*, vol. 27, no. 4, p. 12, 2006.
[6] L. R. Bahl, F. Jelinek, and R. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 179-190, 1983.
[7] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial Intelligence: A Modern Approach*, Prentice hall Englewood Cliffs, vol. 2, 1995.
[8] J. E. Laird, "Research in human-level ai using computer games," *Communications of the ACM*, vol. 45, no. 1, pp. 32-35, 2002.
[9] M. Riedl and V. Bulitko, "Interactive narrative: A novel application of artificial intelligence for computer games," presented at AAAI, 2012.
[10] J. Van Waveren, "The Quake III Arena bot," University of Technology Delft, June 28th, 2001.
[11] R. Coulom, "Monte-Carlo tree search in crazy stone," in *Proc. Game Prog. Workshop*, 2007, Tokyo, Japan, pp. 74-75.
[12] C. Kepner and B. Tregoe, "The rational manager: a systematic approach to problem solving and decision making," Kepner-Tregoe, 1976.
[13] C. Bailey and M. Katchabaw, "An emergent framework for realistic psychosocial behaviour in non player characters," in *Proc. 2008 Conference on Future Play: Research, Play, Share*, 2008, pp. 17-24.
[14] M. Shaker, N. Shaker, and J. Togelius, "Evolving playable content for cut the rope through a simulation based approach," in *Proc. the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2013.
[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," arXiv preprint arXiv: 1312.5602, 2013.
[16] M. Campbell, A. J. Hoane Jr, and F.-H. Hsu, "Deep blue," *Artificial Intelligence*, vol. 134, no. 1, pp. 57-83, 2002.
[17] D. Billings, D. Papp, J. Schaeer, and D. Szafron, "Poker as a testbed for AI research," *Advances in Artificial Intelligence*, pp. 228-238, Springer, 1998.
[18] R. Cook, G. Bird, G. Lünser Unser, S. Huck, and C. Heyes, "Automatic imitation in a strategic context: players of rock-paper-scissors imitate opponents' gestures," in *Proc. the Royal Society B: Biological Sciences*, 2011.
[19] Y. Sato, E. Akiyama, and J. D. Farmer, "Chaos in learning a simple two-person game," in *Proc. the National Academy of Sciences of the United States of America*, vol. 99, no. 7, pp. 4748-4751, 2002.
[20] G. Pozzato, S. Michieletto, and E. Menegatti, "Towards smart robots: Rock-paper-scissors gaming versus human players," in *Proc. Popularize Artificial Intelligence (PAI2013)*, December 2013.
[21] Z. Wang, B. Xu, and H.-J. Zhou, "Social cycling and conditional responses in the rock-paper-scissors game," arXiv preprint arXiv: 1404.5199, 2014.
[22] M. E. Bouton, *Learning and Behavior: A Contemporary Synthesis*, Sinauer Associates, 2007.
[23] C. Wedekind and M. Milinski, "Human cooperation in the simultaneous and the alternating prisoner's dilemma: Pavlov versus generous tit-for-tat," in *Proc. the National Academy of Sciences*, 1996, vol. 93, no. 7, pp. 2686-2689.
[24] S. Valdez, G. Siddayao, and P. Fernandez. "Source code, tools and data for the paper: the modified beat pick algorithm in the first international RoShambo tournament," doi: 10.5281/zenodo.10478.

Sony E. Valdez was born in Agoo, La Union, Philippines, on March 15, 1984. He received his B.S. degree in computer science from Agoo Computer College, La Union, Philippines in 2004, and his M.S. degree in information technology from the University of the Cordilleras in 2010. He is currently studying for his doctorate in information technology at the University of the Cordilleras in the Philippines. He takes interest in the field of artificial intelligence, video games, pokémon and programming.

He has worked in the programming industry for over 11 years and he has been teaching masters and college IT students for the past 3 years. His current research involves developing an artificial intelligence to play in competitive games such as Rock-Paper-Scissors. He is looking to extend the AI research to include boxing games, Pokémans and Street Fighting.

Prof. Valdez is currently the president of Python.PH, Inc, a non-profit organization with the goal of spreading the programming language Python to the country. He is a member of the International Game Developer's Association (Manila Chapter) since 2009.

Generino P. Siddayao was born in Tuguegarao City, Philippines, on April 21, 1980. He received his B.S. and M.S. degrees in information technology from Saint Paul University Philippines (SPUP), Tuguegarao City, Philippines in 2001 and 2004, respectively; He takes interest in systems analysis and programming for various enterprise applications.

He has worked in the academe for over 12 years. He is currently an assistant professor and a university dean for eight campuses of Cagayan State University at Cagayan, Philippines. Current his research interests include artificial intelligence applications in geographical information systems for risk assessment and resource management.

Prof. Siddayao is a member of Philippine Society of Information Technology Educators (PSITE) and currently the President of Council of Deans in Information Technology Educators (CDITE) in Region 02, Philippines.

Proceso L. Fernandez was born in Quezon City, Philippines, on June 19, 1972. He received the B.S., M.S. and Ph.D. degrees in computer science from the Ateneo de Manila University, Quezon City, Philippines, in 1994, 2001 and 2009, respectively. He is currently an associate professor at the same university. He has been a visiting researcher at the Nara Institute of Science and Technology, Nara, Japan, since 2011. He has also been a visiting professor at the Ateneo de Davao University, Davao, Philippines and at the University of the Cordilleras, Baguio, Philippines. He is currently a member of the Computing Society of the Philippines.