Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 631–634

ISBN 978-83-60810-22-4

Minority Game: the Battle of Adaptation, Intelligence, Cooperation and Power

Akihiro Eguchi, Hung Nguyen
University of Arkansas, Fayetteville, AR, U.S.A.
Email: {aeguchi, hpnguyen}@uark.edu

***Abstract*—Minority game is a simulation of a zero-sum game, which has a similar structure to that of a real world market like a currency exchange market. We discuss a way to implement the game and provide a simulation environment with agents that can use various types of strategies to make decisions including genetic algorithms, statistics, and cooperative strategies. The goal of this simulation study is to find the effective strategies for winning the zero-sum game. Results show that both honesty and dishonesty can lead to a player's success depending on the characteristics of the majority of players.**

I. Introduction

THE El Farol Bar problem was proposed in 1994 by W. Brian Arthur [1]. This problem involves inductive reasoning using previous histories of other agents to make their decisions. In this problem, there are *N* people who independently decide every week to go to a bar or not.

The El Farol Bar problem can also be used in market contexts with agents buying or selling an asset at each step. After each step, the price of the asset is calculated by a simple supply and demand rule: if there are more buyers, the market price will be high, and conversely, if there are more sellers than buyers, the market price will be low. With the price high, the sellers do well, and with the price low, the buyers do well. In either way, the minority group always wins.

A variant of the El Farol Bar problem is the minority game, which was proposed by Challet, Marsili, and Zhang [2]. In the minority game, we have an odd total number of players to always produce majority / minority decisions. If the majority of people stay home, the bar becomes "enjoyable" while if the majority of people go to the bar, it becomes "crowded". Therefore, the payoff of the game is to declare the agents who take minority action as winners, the majority as losers.

Another important aspect of the minority game is that this is a zero-sum game, in which the decisions of each player influence those of others, and the total sum of the profit and loss of all players becomes zero. In this paper, we assume the minority game to be a simple economic model and discuss the characteristics of the zero-sum game by implementing the minority game [3].

II. Scoring Method

Agents are limited in their computational abilities, and they can only retain the last *M* game outcomes in their memory. This memory acts like a shift register with the new bit pushing out the oldest bit. Every agent makes his/her next decision based only on these *M* bits of historical data.

Each agent accumulates "capital" reflecting his/her overall score. The agent gets a real point only if the strategy used wins in the next turn. To make the game closer to a real-life stock market, we chose to use our own system of scoring:

If the agent wins a round: *score* = *score* + *nMajority*

If the agent loses a round: *score* = *score* - *nMinority*

with *nMajority* and *nMinority* denoting the number of players in the majority and minority groups, respectively. Because initially all agents receive 0's as their scores, this scoring system ensures that at any time, the sum of all agents' scores stay the same, or that the total score (capital in the real world) is conserved, but its distribution changes after each turn.

III. Implementation

Our simulation is an agent-based model [4] with four different types of agents. Normal agents make decisions using genetic algorithms [5]. Given a sequence of the last *M* outcomes, there are 2^M possible inputs for an agent's decision making. A strategy specifies what the next action is for every sequence of last M outcomes, so there is a total number of 2^{2^M} possible strategies. After every turn, a certain number of strategies with poor performances, calculated based on a virtual score the agent could have been with the strategy, is discarded, and new strategies are randomly generated.

Team agent is a Normal agent who belongs to a team to share their memories to make a team decision without accessing previous decisions of others as done in a previous work [6]. The agents with the higher scores have more weight for their votes by the following formula:

$$R = \sum_{i=1}^{nTeamMembers} \frac{s_i - min}{max - min} \times r_i$$

where s_i is the score of agent i; and r_i is the response of the agent (-1 or 1). If $R > 0$, the team's decision is to stay, and vice versa.

Additionally, we assign each Team agent a percentage of loyalty towards their team. We have Team agents with two types of loyalty: the first is to tell a lie about his intention when voting for the team decision (type 1), and the second is to do the opposite of the team decision (type 2).

Super agent is another agent who makes decisions based on a certain number of previous market results, which weigh more on the most recent history. It calculates the probability of the Super agent to go to the bar in the next turn:

$$P = 100 - \left(\sum_{i=1}^{nHists} i^2 \times HistPercentStay[i]\right) / \sum_{i=1}^{nHist} i^2$$

This is a simple way to predict the market, and it is based on the assumption that the number of agents that go and the number that stay will converge in the long run as they learn.

IV. Data Structure

To conserve space, instead of using Java's boolean to store the histories of agents, we use short instead. Each boolean costs 1 byte, so for an agent with 10 history entries, it costs 10 bytes to save his history, which seems like a small number. However, if each agent has 12 strategies, the total number of bytes used to store all possible histories for a normal agent's strategies is over 200 MB for 1501 agents. Therefore, instead of using 10 boolean values to indicate whether the agent has won or lost the past latest 10 turns, we chose to use one bit to store the outcome of one turn, so in total we only need 1 bit instead of 1 byte to store the history. Because the maximum number of history entries an agent can store is 10, we use one Java's short value to store all the history entries. Thus, instead of 10 bytes, we only need 2 bytes for each agent. In total, we only need 3 KB instead of 15 KB. Now using the same approach for histories stored within each strategy (not within each agent), the total number of bytes we need to allocate for the strategies is just over 55 MB. This is a good upper bound for memory consumption, compared to 200 MB with Java's boolean.

Since each agent's history is a short, we can just store each strategy's responses as a boolean array and use the agent's history as the index for that array. In the end, we are able to reduce the number of bytes allocated for all strategies further to about 18.5 MB for the same configurations.

In addition, instead of using arrays of booleans to store the responses of Normal strategies, we chose arrays of integers, so the boolean value in the previous implementation is now represented by only one bit. Since one boolean value consumes 8 bits (1 byte), this method of representing one boolean value using 1 bit thus reduces the memory consumption 8 times. As a result, we increase the maximum number of Normal agents from 1,600 to 10,000 on our laptops. Therefore, our final design consumes a total of about 2.3 MB, or almost 1/100 of the initial 200 MB.

V. Simulation

A. Normal agents

Suppose there are 1001 Normal agents with the memory size of 6 turns in the simulation, Figure 1 shows the graph produced when it runs for 3650 turns. In the graph on the left side, the red curve represents the best score of all agents in the simulation while the blue curve represents the absolute value of the worst. The bar graph on the right shows the distribution of the scores of each agent. The average score of all agents, which is zero, is denoted as a red vertical line.

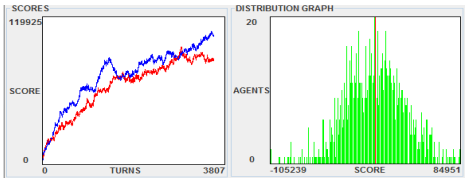

Fig 1. Graph of Normal agents

The distribution graph shows a well-balanced bell curve that shows a normal distribution of the agents' scores. The genetic algorithm, which the Normal agents use, optimizes the winning probability of each agent, so supposedly each agent fine tunes their decision-making strategy and increases their score as time goes on. However, because of the characteristics of the zero-sum game, all agents cannot win at the same time. This configuration shows the characteristics of a zero-sum game very well.

B. Normal agents and Super agents

We ran the simulation again with 1001 agents for 3650 turns, but with 501 Normal agents and 500 Super agents who use statistics based on market history rather than their past decisions.

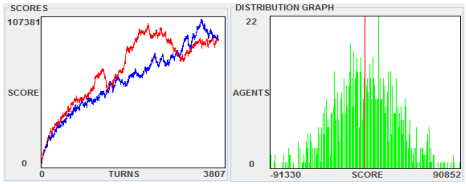

Fig 2. Graph of Normal agents and Super agents

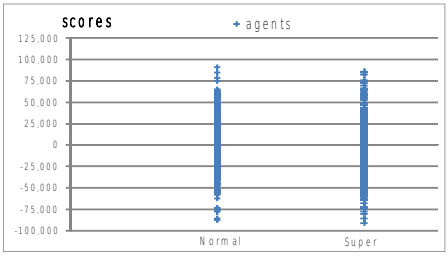

Fig 3. Comparison of Normal and Super scores decomposed

Interestingly, similar to the previous configuration, we observed a well-balanced normal distribution (Figure 2). In addition, the distribution of each type of agent is very similar to each other (Figure 3). This suggests that the bell curve in Figure 2 is the sum of two smaller and almost identical bell curves. Thus, it shows that simple statistics based on a market history can make wining decisions as good as genetic algorithms can.

C. Normal agent and loyal Team agents

Another interesting configuration is when Team agents are 100% loyal to their teams. The participants in this configuration consist of 501 Normal agents and 5 teams, each of which consists of 100 members.

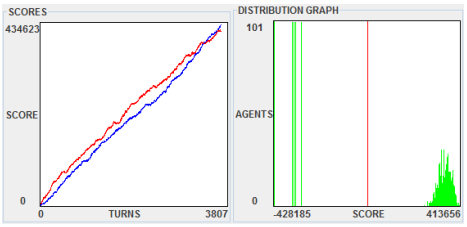

Fig 4. Graph of Normal agents and team agents

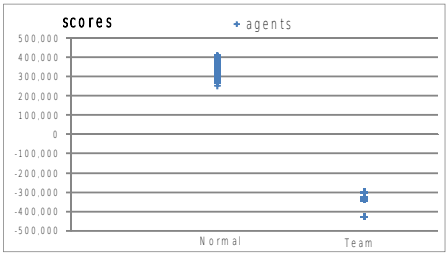

Fig 5. Comparison of Normal and Team scores

Team agents perform far more poorly than Normal agents since they all do exactly the same thing. As this is a minority game, this type of Team strategy is the worst of all strategies. Figures 4 and 5 also show how the scores of theses two types of agents are highly independent of each other.

D. All agents in one simulation

In the next two configurations, we ran the experiments with 501 Normal agents, 5 teams of 100 agents each, and 500 Super agents for 3650 turns. Team agents now have their loyalties randomly generated for both types.

1) With disloyal Team agents: liars (Type 1)

Type 1 disloyal Team agents lie about their intentions.

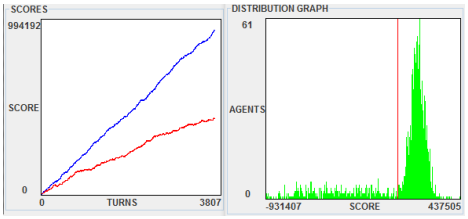

Fig 6. All agents – Team agents use team strategy 1

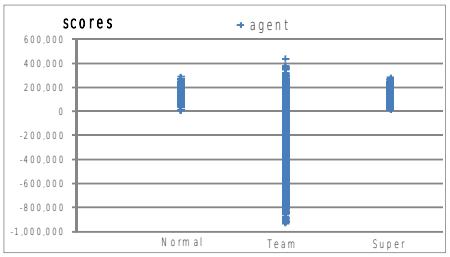

Fig 7. Comparison of agent types

Using Figure 7 and simulation B, we can conclude that the bell curve to the right of the red average line in Figure 6 is the sum of Normal and Super agents. Their averages are much better than those of Team agents (Figure 7). Team agents' scores are scattered, and there is a very strong negative correlation between score and loyalty. About 20% of the

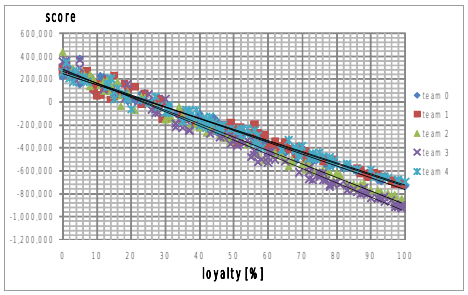

Fig 8. Plotting of Team agents who use team strategy 1

Team agents who are the most disloyal perform above the zero mark.

2) With disloyal Team agents: perverse fellows (Type 2)

In this case, type 2 disloyal agents do the opposite of their team's resolution. We predicted that the team decisions affect the majority of the members, so the ones who do not follow the team's agreement are more likely to win.

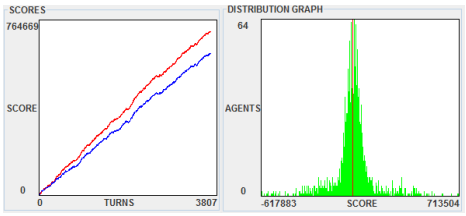

Fig 9. All agents – Team agents use team strategy 2

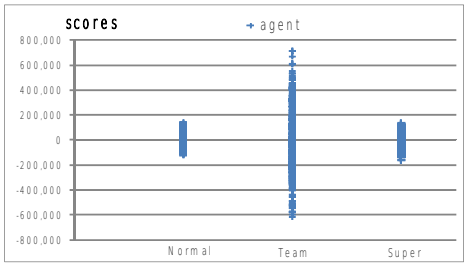

Fig 10. Comparison of agent types

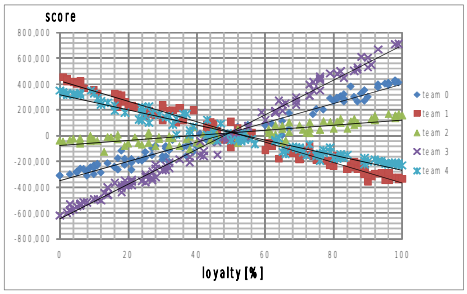

Fig 11. Plotting of Team agents who use team strategy 2

To our surprise, the negative correlation between loyalty and score as in type 1 disloyalty is not true anymore. There is either a positive or negative correspondence between loyalty and score for members of the same team. The ones who betray their team 50% of the time do not lose or gain anything in the game (Figure 11). In contrast to Figure 7, Figure 10 shows how the Team agents, taken as a whole, perform just as well as the Normal and Super agents. However, Team agents' scores are more spread out. When viewing both Fig-

ures 7 and 10, we can recognize how the best Team agents win over the Normal and Super agents, and even make their scores shift down dramatically. This helps "even out" the field, so the score distribution goes back to the bell curve shape (Figure 9).

E. The final showdown of Team agents

To see how the two types of Team agents would compete with each other, in this final simulation, we have eight teams, whose loyalties are randomly generated, with 100 agents each: the first four are type 1 and the other four are type 2.

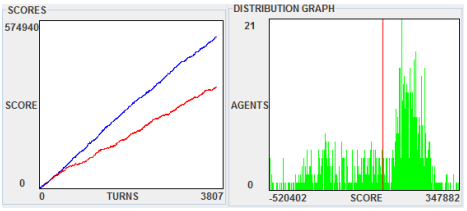

Fig 12. Two types of Team agents

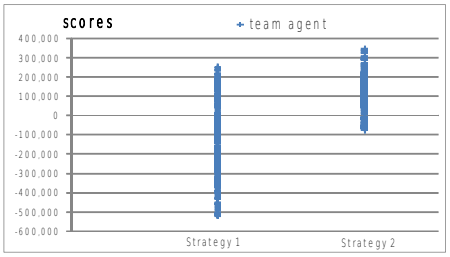

Fig 13. Comparison of team strategies

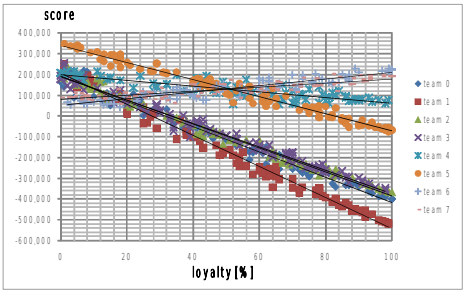

Fig 14. Plotting of two types of Team agents

Type 2 outperforms type 1 both absolutely and on average. Furthermore, the score distribution of teams following type 2 is more balanced (Figure 14), and the difference between the highest and lowest scores is less than those of teams following type 1 (Figure 8). However, because teams with type 2 "steal" from teams following type 1, the total score distribution is skewed to the right.

VI. Observations

If we take a look at the wealth distribution in the real world, we can see a similar trend to our simulation. According to a report by Dr. Zhu Xiao Di, in the United States in 2004, the top 25 percent of households owned 87 percent of the wealth in the country whereas the bottom 25 percent of households owned nothing [7]. This is illustrated in Figures 7 and 8, which show the scores of loyal team agents who are "bullied" by both their disloyal teammates and by Normal and Super agents. From the results, if you are a Normal agent, then you will likely live an average life. However, if you are a Team agent, i.e., more prone to be positively or negatively influenced by other people, your life could be extreme in either way. At first, we assumed that being disloyal to the society that one belongs to would be the only way to win the game, but the simulation showed that this is not always the case. When the honest and loyal are the minority, they win against the tricksters and treacherous majority; and vice versa. Therefore, we successfully showed that similar to the real world, any organization could be extremely successful either by being honest or dishonest depending on its environment.

VII. Conclusion

In this paper, we discussed a way to implement a variant of the minority game. In order to deal with a large number of participants in the simulation, we introduced several ways to optimize the architecture. Then, we ran the simulator with several strategies to observe the results. Because of the characteristics of the game, the genetic algorithm produced a normal distribution. We also showed how a genetic algorithm is comparable to statistical analysis of the game. Since this is a minority game, team decisions play an important part. If all Team agents are honest, they are likely to perform worse than Normal agents. Then, we showed how the three types of agents would perform together in the same game with disloyal Team agents who lie to their teammates (type 1), or who do the opposite of their team's decision (type 2). The observation is that type 2 helps Team agents as whole: it makes the best Team agents better than the best of Normal and Super agents and narrows the gap between the worst of team agents and the worst of the other two types. There also exist both positive and negative correlations between loyalty and score with type 2 while only negative correspondence is observed with type 1. The last simulation illustrates how type 2 is more effective than type 1. This also shows that to win this game, both loyal and disloyal Team agents can win. The losers are the ones with opposite characters of their own teams.

References

[1] W. B. Arthur, "Inductive Reasoning and Bounded Rationality," in *American Economic Review*, vol. 84, pp. 406 - 411, 1994.

[2] D. Challet, M. Marsili, and Y.-C. Zhang, *Minority Games*, Illustrated edition. Oxford Univ Press, 2005.

[3] G. W. Greenwood and R. Tymerski, "A game-theoretical approach for designing market trading strategies," in *Proc. 2008 IEEE Conf. Computational Intelligence and Games*, pp. 316-322, 2008.

[4] A. Chakraborti, I. M. Toke, M. Patriarca, and F. Abergel, "Econophysics: Empirical facts and agent-based models," 2009.

[5] Marko Sysi-Aho, Anirban Chakraborti, and Kimmo Kaski. "Biology helps you to win a game," in *Proc. Unconventional Applications of Statistical Physics Conf.*, vol. T106 of *Physics Scripta T-series*, pp. 32-35, 2003.

[6] T. Kalinowski, H.-J. Schulz, and M. Briese, "Cooperation in the Minority Game with local information," *Physica A: Statistical Mechanics and its Applications*, vol. 277, issues 3-4, pp. 502-508, Mar. 2000.

[7] Z. X. Di, "Growing Wealth, Inequality, and Housing in the United States," Joint Center for Housing Studies, Harvard Univ., Cambridge, MA, W07-1, 2007.