Game Theory and Mechanism Design

Imperial College London

Department of Design Engineering

Minority Games

Joseph Johnson

CID: 01850831

GitHub Repository

Contents

1 Definition

1.1 What is a Minority Game?

Minority games are a field of game theory, that is representative of many real world situations such as financial markets or traffic routing. Players, or agents, in the game are challenged with trying to predict the trends and decisions made by the other players in order to best select an option, and thus maximise their respective payoff.

1.2 Components of a Minority Game

1.2.1 Rounds, $\{R \in \mathbb{Z} \mid R \geq 1\}$

A game takes place over R rounds.

1.2.2 Agents, $\{A_i \mid i = 0, 1, 2, \ldots, n\}$

There exists n number of agents in the simulation, who compete to maximise their payoff each round.

1.2.3 Options, $\{O_i \mid i = 0, 1\}$

Agents can choose between two options each round, 0, or 1.

1.2.4 Strategies, $\{S_i \mid i = 0, 1, 2, \ldots, p\}$

In order to choose O, agents can select from p strategies, where each S_i is a different method of predicting the best choice such as trying to identify a pattern in overall agent behaviour or randomly selecting an option.

1.2.5 Minority Threshold, $\{T \in \mathbb{R} \mid 0 < T < 1\}$

A threshold, T, represents the proportion of agents that can fit within the minority group. For example, if the minority threshold is 0.3, then if less than or equal to 30 percent of agents choose option 0 then they will be rewarded. However, if more than 30 percent choose option 0, then agents who choose option 1 are rewarded instead.

1.3 Forms of a Minority Game

1.3.1 Single Shot Normal Form Game

When $R = 1$, the game is considered a single shot game. Agents only have one chance to make a decision, and this decision is made without any prior knowledge of overall agent behaviours.

1.3.2 Repeated Form Game

In a more complex game, where $R > 1$, agents seek to maximise their cumulative payoff over multiple rounds, which is done by trying to maximise payoff at each round of the game. This setup allows for more complex strategies to be applied, known as inductive strategies, which apply experience gained from prior rounds to improve an agents odds of maximising their payoff.

2 Theoretical analysis

2.1 Nash Equilibrium of a Static Normal Form Game

Before any computational simulation was performed, the game was considered through the theoretical approach of examining the single shot game.

This scenario can be simplified as shown by the normal form game in fig. 2.1, where any given agent, A_i, can be modelled to play against the rest of the agents, $A - A_i$. If A_i is to choose the same option as the rest of the agents, then they will receive no reward, however, if they can select the opposite to the rest of the agents, then $A - A_i$ will receive a payoff of 1.

In this game, there occurs two Nash Equilibrium, at $A_i = 0, A - A_i = 1$, and $A_i = 1, A - A_i = 0$. These exist as Nash Equilibrium as from these positions in the normal form game shown in fig. 2.1, by moving laterally neither side can increase their payoff.

This duality of Nash Equilibria creates the uncertainty of the game, as akin to the problem posed in the 'Battle of the Sexes' game theory problem. With all agents playing the same game, trying to guess against the majority, there is no clear winning strategy and as such, we can expect a scenario that is challenging to predict.

$$\begin{array}{cc|c|c|}
 & & \multicolumn{2}{c}{A - A_i} \\
 & & \mathbf{0} & \mathbf{1} \\
\hline
\multirow{2}{*}{A_i} & \mathbf{0} & \overline{(0,0)} & \underline{\overline{(1,0)}} \\
\cline{3-4}
 & \mathbf{1} & \underline{\overline{(1,0)}} & \overline{(0,0)} \\
\hline
\end{array}$$

Figure 2.1: Normal Form Representation of a Minority Game

3 Implementation and simulation of static game version

3.1 Algorithm

3.1.1 Agent Class

Agents are modelled using the `Agent()` class. Calling the `Agent.decide()` function, the agent will return either `0` or `1`. For the static game, as there is no history, this decision is made using a random function weighted by T, as shown in listing C.1.

3.1.2 Game Class

Minority Games are modelled using the `MinorityGame()` class. When initialised, the `Agent` objects are created, along with a `history` array, used to store the games history, and a number of additional variables used to track metrics over the course of the game such as `strategy_decision_count` and `strategy_counts_per_round`.

3.1.3 Simulation Class

The simulation class is used to run multiple games, allowing for plots to be created that show general trends across many, independent games. This is useful for determining trends in the results due to the randomness of a single game.The Simulation class also includes functions to create data visualisations, as used in 3.2.

3.1.4 Simulation Parameters

The algorithm can be setup using a number of configurable constants. These can be changed in order to define how the simulation is run such as the minority threshold , T, or the number of agents , n, and used to find a balance between finding meaningful insights while maintaining a suitable runtime. These parameters are shown in table B.1.

3.2 Results & Analysis

The simulation was run with 100 games, each with 101 agents in each, and a minority threshold of 0.3. The distribution of the percentage of agents selecting 0 can be seen in fig. 3.1. From this, it is shown that the mean percentage of agents selecting the 0 option is equal to the minority threshold, as expected from the weighted average formula.

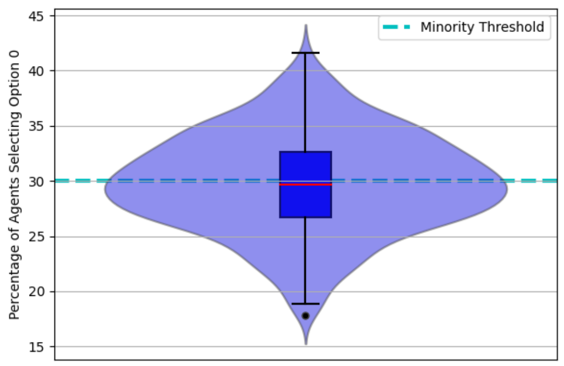


Figure 3.1: Percentage of Agents Selecting Option 0 Across All Games

4 Implementation and simulation of repeated game version

4.1 Algorithm

4.1.1 Agent Class

As a repeated game without inductive strategies, the agents logic remains largely the same as 3.1.1. A new array used to track the agents decisions over time was implemented, and is used to produce plots as seen in 4.2.

4.1.2 Game Class

The game class receives the biggest change from 3.1.2. As shown in table B.2, the `num_rounds`, R, can be configured. The game will iterate over each round allowing a repeated game to occur.

4.1.3 Simulation Class

The simulation class remains unchanged from 3.1.3, apart from additional plotting and analysis functions.

4.1.4 Parameters

For the repeated setup, additional parameters are implemented, as shown in table B.2, which are used in addition to those detailed in table B.1.

4.2 Results & Analysis

The repeated game simulation was run with a minority threshold of 0.3, over 20 games, each with 200 rounds and 101 agents within each game. The average number of agents choosing option 0 fluctuated around 0.3, the same as the minority threshold, as shown in fig. 4.1. The spread of agents selecting option 0 is comparable to fig. 3.1 - which is to be expected as since the agents do not adapt their strategies over time, the probability of an agent selecting the 0 option in any round of the repeated game simulation is the same as that of an agent in the single round of the single shot simulation.

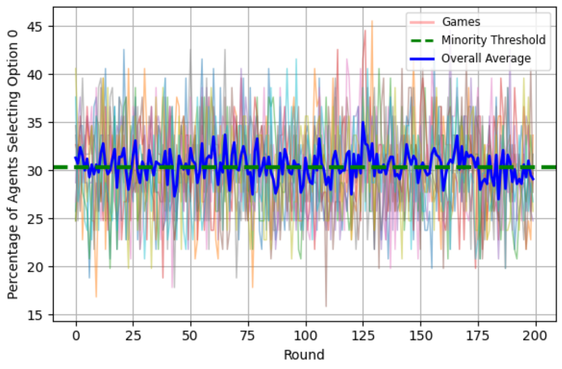


Figure 4.1: The percentage of Agents selecting option 0 over time

The probability distribution of agent scores across the games followed a normal distribution, as shown in fig. 4.2. The mean score of 92.31 shows that on average, an agent will score a point in 46% of the rounds that they play.

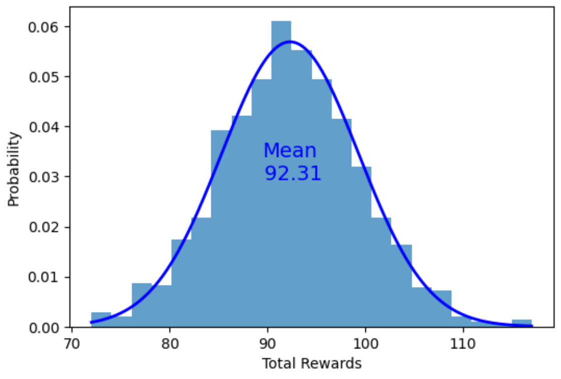


Figure 4.2: Probability distribution of scores across all agents in all games

I explored the impact of varying the minority threshold, by running multiple simulations with different values for T and comparing it with the average rate that agents received the payoff, from which the results of which can be seen in fig. 4.3. This shows a symmetry around 50%, which is to be expected as a minority threshold of x is equivalent to a threshold of $1 - x$.

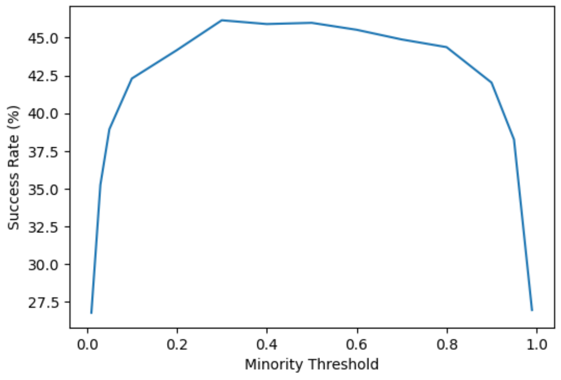


Figure 4.3: Average success rates of agents over varied minority thresholds

5 Proposal of Inductive Strategies

5.1 What is an Inductive Strategy?

An inductive strategy is a way for an agent to make their decision based off of their experiences from prior rounds. These can vary in complexity, as demonstrated in 5.2, ranging from simple methods such as selecting the same option as that which was rewarded last time, to complex algorithms that learn and develop strategies over many rounds. Agents have access to a broad array of strategies, leading to a need for an overarching strategy to select between these, such as a softmax selection process as detailed in 5.3.

5.2 Proposed Inductive strategies

5.2.1 Repeat Last

The simplest inductive strategy, in which the agent will choose the same option that maximised payoff in the last round. This is shown in fig. A.1

5.2.2 Inverse Last

In contrast to the Repeat Last strategy, Inverse Last will choose the option that did not maximise payoff in the last round. This is shown in fig. A.2

5.2.3 Genetic

The genetic strategy is inspired by evolution, in which successful genes crossover in order to produce a more successful offspring overtime. Like in evolution, randomised mutations also occur in order to encourage exploration over time. This process can be seen in fig. A.3.

5.2.4 Bayesian

The Bayesian strategy uses a probabilistic model to determine O_i, based off of its confidence in O_0 receiving the pay off. It's beliefs are updated each round, based off of its experiences. This process is shown in fig. A.4

5.2.5 Adaptive

The adaptive strategy considers the average choice made over past rounds, and compares the mean of winning choices to T. This can be seen in fig. A.5

5.2.6 Market-Based

In a Market Based strategy, the agent assigns a price to each O_i, which is updated each round based on how often each is chosen. Each round, the agent plays O_i with the lowest market value, as seen in fig. A.6.

5.2.7 Pattern Recognition

The pattern recognition strategy attempts to find repeating patterns in the games history, and uses this to predict which O_i will receive the payoff at each round. This process can be seein in fig. A.7.

5.3 Switching between strategies

As all agents have access to all strategies, they must have a means of selecting which one to use. For this, each agent will generate its own set of scores for each strategy, based off of performance, faliure count, and time since last change. A softmax function then converts these to a probability distribution, from which the strategy is selected for use. This process is run each round, allowing its strategy to update as the game is played out. This process can be seen in fig. A.8.

6 Implementation and Simulation of Inductive Game Version

6.1 Algorithm

6.1.1 Agent Class

The agent class is greatly changed to accommodate inductive strategies. Additional logic is implemented in order to perform the strategies proposed in 5.2, along with the logic for switching between strategies as detailed in 5.3.

6.1.2 Game Class

The simulation class remains largely unchanged from 4.1.2.

6.1.3 Simulation Class

The simulation class remains largely unchanged from 4.1.3, apart from additional plotting and analysis functions.

6.1.4 Parameters

For the inductive setup, additional parameters are implemented, as shown in table B.3, which are used in addition to those detailed in table B.1 and table B.2. Of note is the `exploration_rounds` parameter, which determines the number of rounds from which agents randomly select their strategies in order to build up a baselien for each when evaluating under the softmax algorithm.

6.2 Results & Analysis

To begin with, I ran the simulation of 10 games, each with 101 agents, 200 rounds, minority thresholds of 0.3, 10 exploration rounds, and memories of 10 rounds. This established a baseline from which the impact of parameter variance can be explored.

The number of agents selecting option 0 over time are shown in fig. 6.1. In rounds where $y < T$, agents choosing 0 will receive the payoff - likewise when $y >= T$, agents choosing 1 will receive the payoff. During the exploration rounds, there is a slight preference towards option 1, however this appears to converge towards the average percentage of agents selecting option 0 being the same as the minority threshold, in this case 0.3 or 30%.

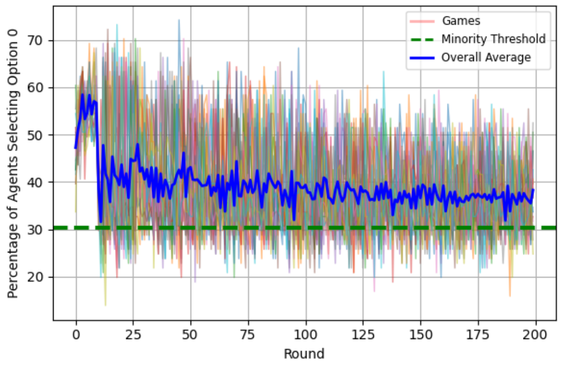


Figure 6.1: The percentage of Agents selecting option 0 over time

As shown in fig. 6.2, the vast majority of agents select the weighted random strategy, a non-inductive approach as was used in the static and repeated game - although it is of note that agents are inductively selecting this approach over others. The convergence towards the minority threshold as shown in fig. 6.1 can be explained partially by this mass adoption over time of the weighted random strategy, as the if 100% of agents used it, it would average around the threshold, just as in the repeated non-inductive game shown in fig. 4.1. However, of the other strategies, there also appears to be strong preference of some over others. Repeated last is the next most popular, followed by genetic and market based respectively.

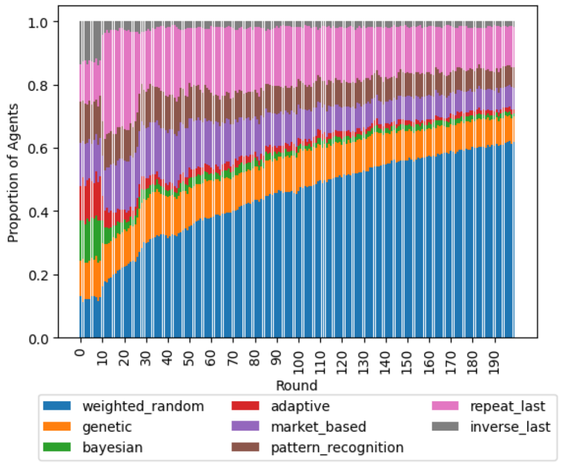


Figure 6.2: The Distribution of Strategy Choices over Time

The preference for the weighted random and repeat last strategies are evident in fig. 6.3, being the most popular to switch to and from. They both show significant switches to and from in comparable amounts, suggesting that many agents will fluctuate between the two over the course of the simulation, under the influence of the softmax selection strategy.

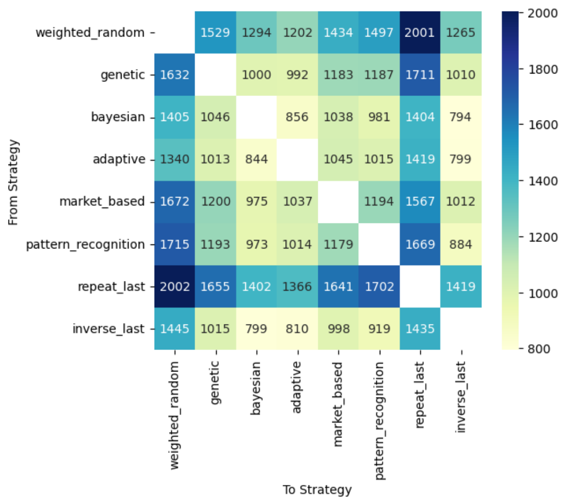


Figure 6.3: Strategy switching heatmap across all games in the simulation

This selection distribution correlates well with the performance of each strategy, as shown in fig. 6.4. This shows that the softmax process is an effective way of choosing successful strategies.

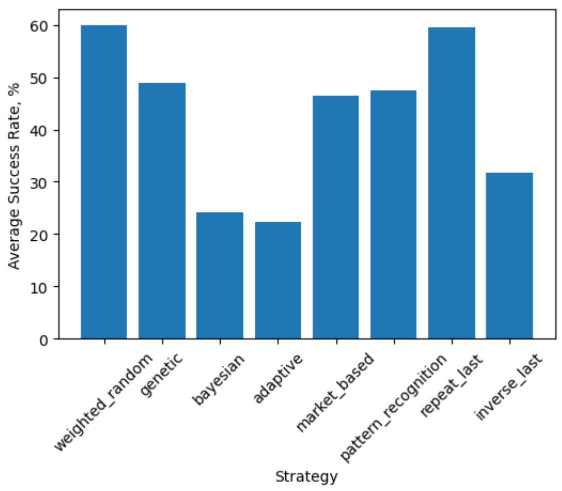


Figure 6.4: Average success rates for each strategy

The distribution of the total rewards of agents over all games follows a normal distribution as shown by the comparison between the histogram and overlaid normal distribution curve in fig. 6.5. This behaviour is akin to the prior repeated simulation results shown in fig. 4.2, although with a higher mean of 103.92, equating to an average success rate of 52%/.

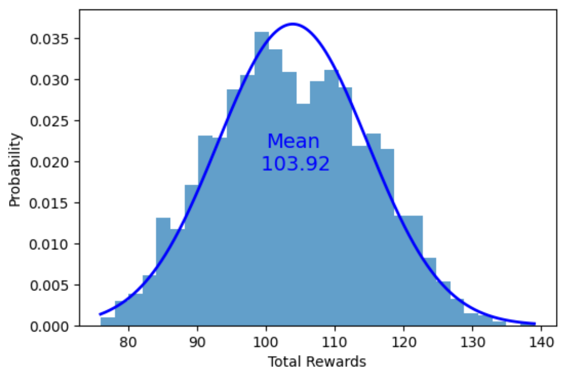


Figure 6.5: Distribution of scores across all agents in all games

Again, as in 4.3, the minority threshold was varied in order to assess the impact this had on the average success rate of an agent, the results of which can be seen in fig. 6.6. This showed an interesting difference between the behaviours of the inductive and non-inductive repeated games. In the case of an inductive game, as the threshold approached 0.5, it decreased; the opposite behaviour occurred in the non-inductive game.

In order to better understand this, a plot of the strategy choices over time was again created, this time with a minority threshold of 0.05, the results of which can be seen in fig. 6.7. This shows that as the threshold gets further from 0.5, weighted random and repeat last become even more popular. As the average success rate is so high, option 1 must be winning the vast majority of the time, suggesting that the majority of agents tend towards selecting option 1, while a small but greater than the majority threshold amount choose option 0.

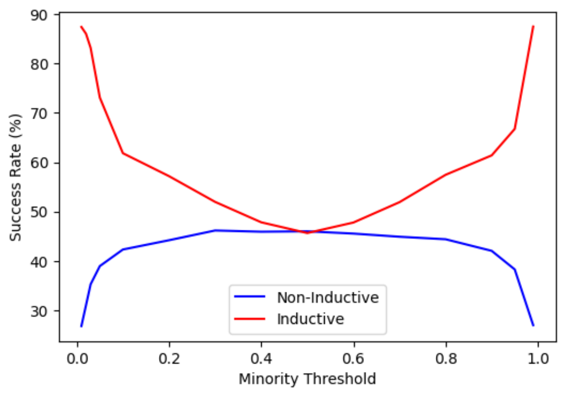


Figure 6.6: Average success rates of agents in an inductive and non-inductive game over varied minority thresholds

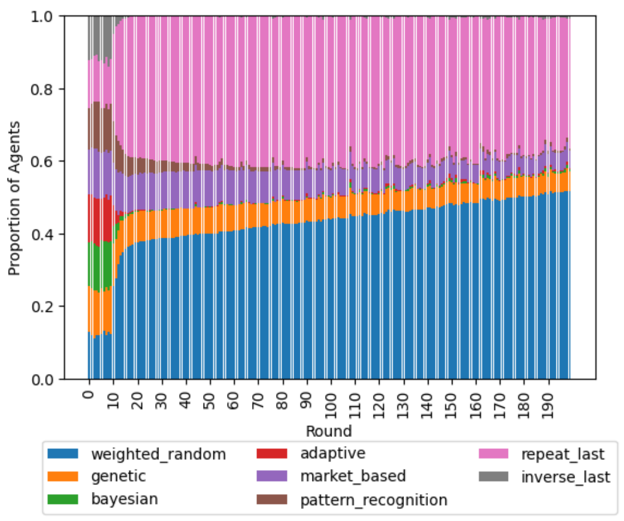


Figure 6.7: The Distribution of Strategy Choices over Time, $T = 0.05$

7 A minority game in the real-world

In the real world, minority games can be found in a broad number of scenarios. One such example is in Formula One. Game theory concepts can be applied to a broad range of aspects in the sport ranging from the engineers who aim to maximise payoff by designing a vehicle that taxes advantage of competitor team weaknesses, while not falling susceptible to their strengths, to the drivers who try to predict the behaviour of all other drivers around them throughout the race, such as when deciding when to attempt an overtake. Overall, in each of these games, both teams and drivers are seeking to maximise their points, which are awarded respective of finishing position in each race. In order to maximise their scores, they seek to minimise their average lap time across all laps of the race.

One such scenario where minority games exist is in the selection of a pitting strategy. Drivers can choose to pit once, or twice. This results in a minority game setup where the minority is advantaged as such:

- If the majority of drivers choose to pit twice, those who do not will be able to easily overtake those in the pits, giving a positive payoff through reduced race time.
- If the majority of drivers choose to pit once, then those who pit twice have the advantage of being able to drive faster as they are able to degrade three overall sets of tyres instead of two, giving the payoff of reduced race time.

In the setup of the 2024 race season, there would exist 20 agents *(drivers)*, and the game would be played over 24 rounds *(races)*. However, in this scenario there are additional factors to consider that extend beyond the scope my simulations:

- Each round may have a differing minority threshold, due to differing track or weather conditions.
- While there are 20 drivers, they work in pairs across 10 teams. This leads to coordinated strategies between team members who aim to maximise both their individual and net team payoff.
- Payoff varies for different teams depending on the speed of their pit crews
- The time that drivers choose to pit also has impact, as too many pitting at once causes congestion in the pit lane. In this way, each lap can be considered a minority game too, where drivers can choose to pit or not to pit.

While the payoff of the pit stop strategy does not directly reward drivers with points, it contributes heavily to their overall race performance when coupled with other key factors such as driver performance.

A Inductive Strategy Flow Diagrams

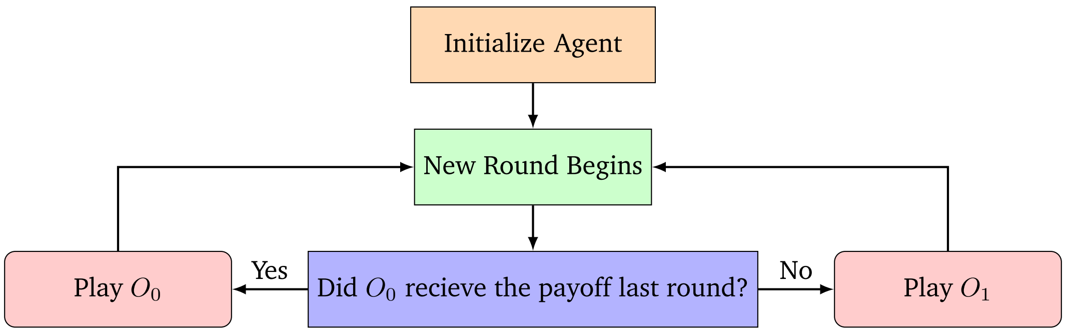


Figure A.1: Flowchart for the Repeat Last Strategy

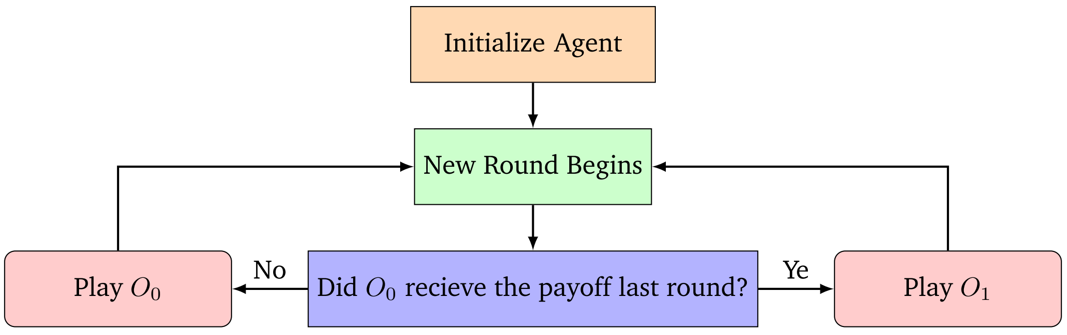


Figure A.2: Flowchart for the Inverse Last Strategy

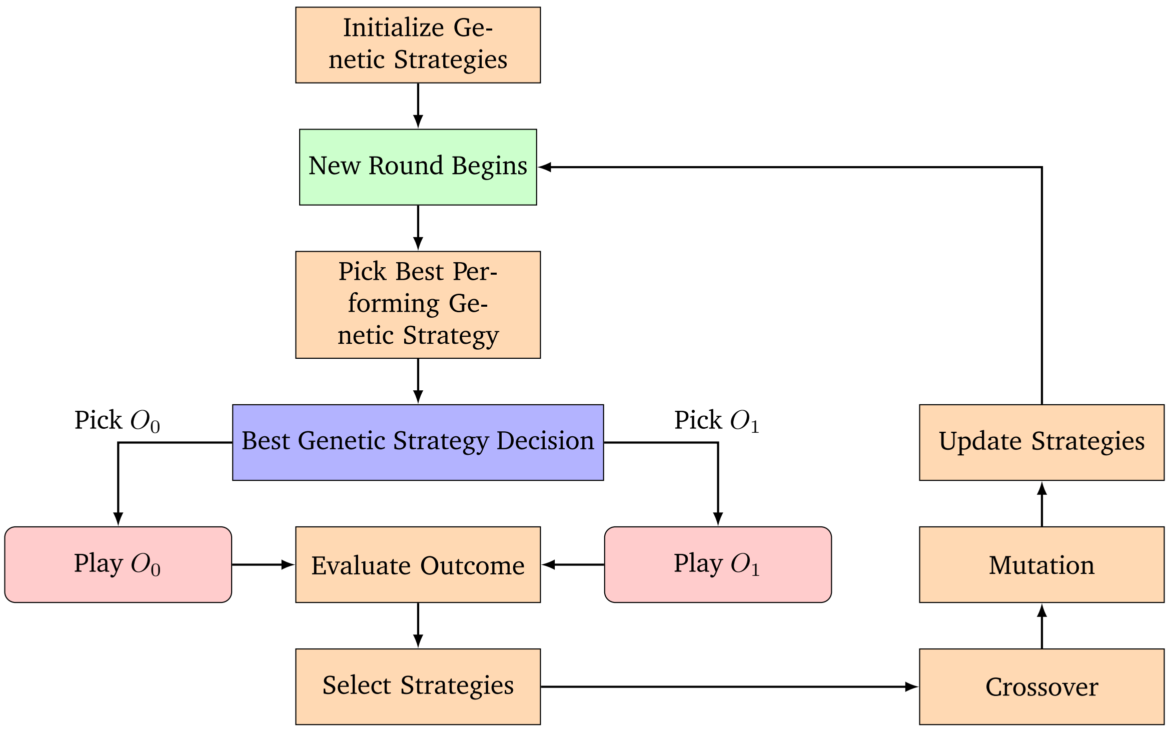


Figure A.3: Flow Chart To Show Genetic Algorithm

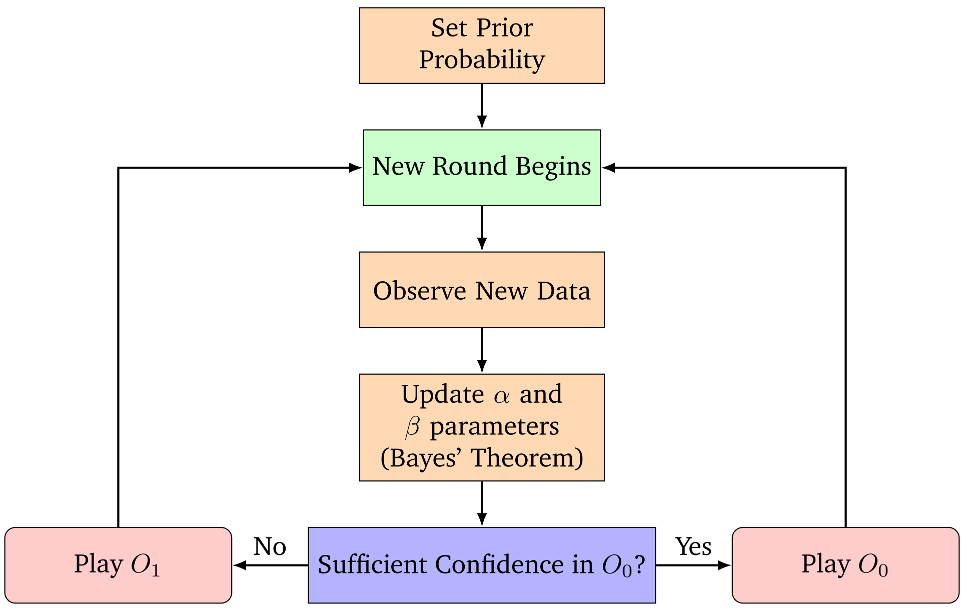

Set Prior Probability
New Round Begins
Observe New Data
Update α and β parameters (Bayes' Theorem)
Sufficient Confidence in O_0?
No
Yes
Play O_1
Play O_0

Figure A.4: Flowchart for the Bayesian Strategy

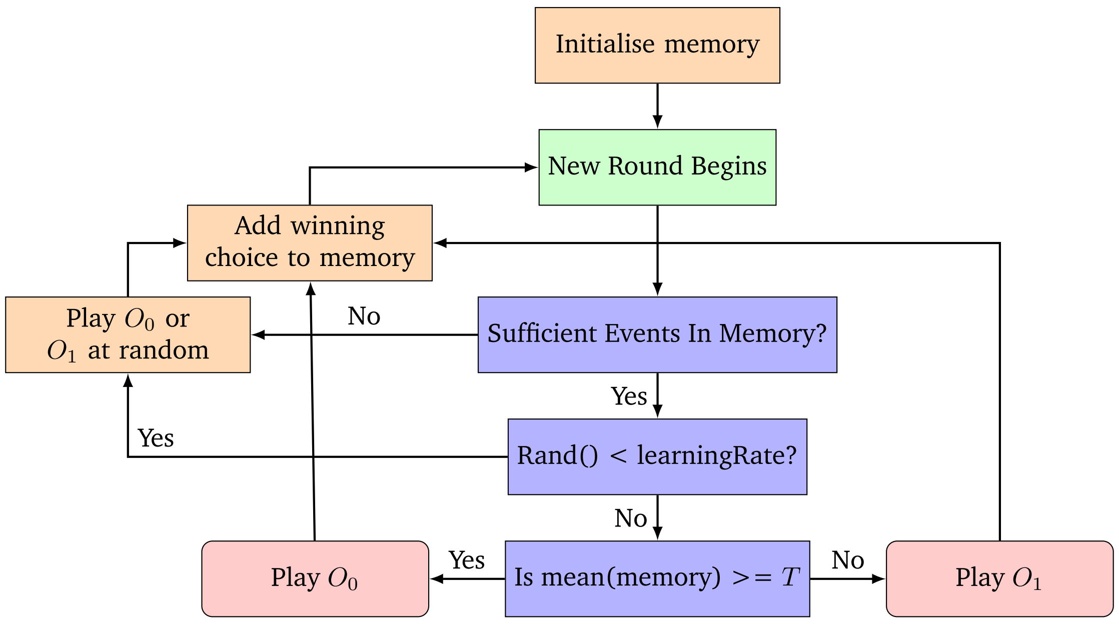
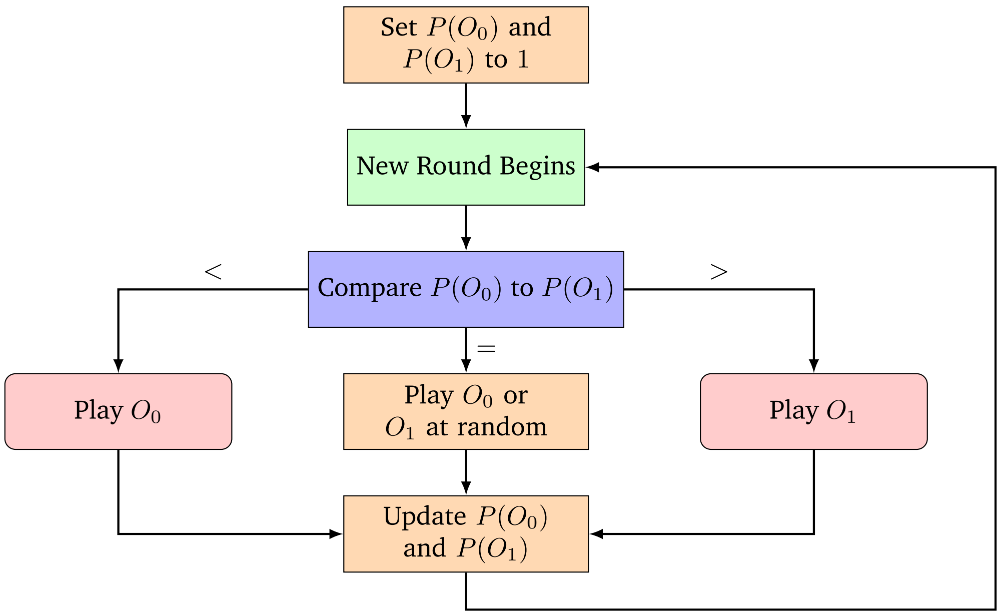


Figure A.5: Flowchart for Adaptive Strategy



Set $P(O_0)$ and $P(O_1)$ to 1
New Round Begins
Compare $P(O_0)$ to $P(O_1)$
$<$
$>$
$=$
Play O_0
Play O_0 or O_1 at random
Play O_1
Update $P(O_0)$ and $P(O_1)$

Figure A.6: Flowchart for the Market-Based Strategy

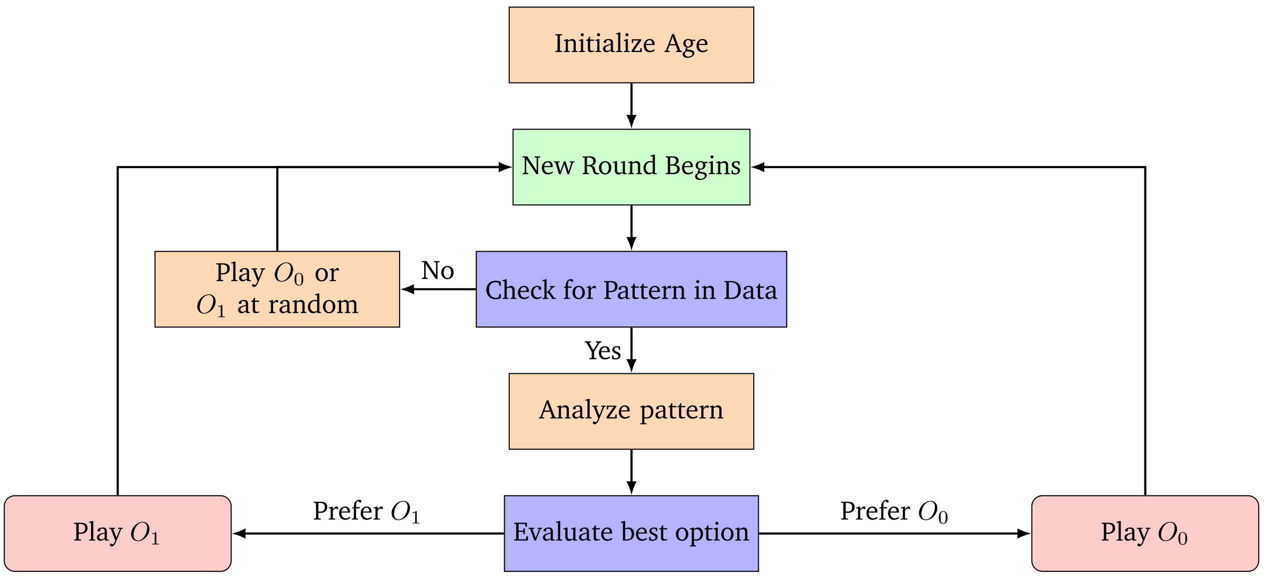


Figure A.7: Flowchart for the Pattern Recognition Strategy

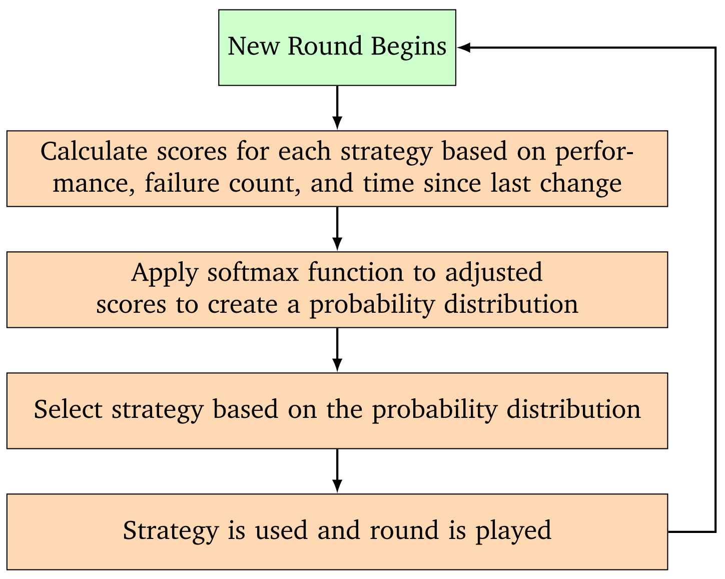

New Round Begins
Calculate scores for each strategy based on performance, failure count, and time since last change
Apply softmax function to adjusted scores to create a probability distribution
Select strategy based on the probability distribution
Strategy is used and round is played

Figure A.8: Flowchart of the Softmax Selection Process

B Simulation Parameters

Parameter	Description	Complexity
`num_agents`, $\{x \in \mathbb{Z} \mid x \geq 3\}$	The number of agents that exist within each game.	$O(N)$
`num_games`, $\{x \in \mathbb{Z} \mid x \geq 1\}$	The number of games ran within the simulation. Multiple games are ran so that averages can be taken across them to determine trends.	$O(N)$
`minority_threshold`, $\{x \in \mathbb{R} \mid 0 < x < 1\}$	The value for the minority threshold, T	$O(1)$

Table B.1: Algorithm Parameters for Static Game

Parameter	Description	Complexity
`num_rounds`, $\{x \in \mathbb{Z} \mid x \geq 1\}$	The number of rounds that each game iterates over. Note that using 1 will run a static game.	$O(N)$

Table B.2: Additional Algorithm Parameters for Repeated Game Version

Parameter	Description	Complexity
`memory_size`, $\{x \in \mathbb{Z} \mid x \geq 1\}$	The number of rounds that inductive strategies can remember. Limited to manage runtime due to exponential complexity.	$O(N^2)$
`exploration_rounds`, $\{x \in \mathbb{Z} \mid x \geq 0\}$	The number of rounds in which agents randomly select strategies in order to build a foundational view on strategy effectiveness.	$O(1)$

Table B.3: Additional Algorithm Parameters for Inductive Game Version

C Code Snippets

```
random_number = random.uniform(0, 1)
if random_number > minority_threshold:
    return 1
else:
    return 0
```

Listing C.1: Weighted Random Strategy