論文 / 著書情報
Article / Book Information

Title	A New Model for Bias-Generating Agent-Based Simulation and Its Application to Election Systems: Allowing Agents to Make Mistakes for a Reason
Authors	Jiateng Pan, Atsushi Yoshikawa, Masayuki Yamamura
Citation	Mathematical Problems in Engineering, Volume 2022, ,
Pub. date	2022, 4
Creative Commons	Information is in the article.

Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 4010945, 12 pages
https://doi.org/10.1155/2022/4010945

Research Article

A New Model for Bias-Generating Agent-Based Simulation and Its Application to Election Systems: Allowing Agents to Make Mistakes for a Reason

Jiateng Pan, Atsushi Yoshikawa, and Masayuki Yamamura

School of Computing, Tokyo Institute of Technology, Tokyo, Japan

Correspondence should be addressed to Jiateng Pan; pan.j.aa@m.titech.ac.jp

Received 26 January 2022; Accepted 7 March 2022; Published 13 April 2022

Academic Editor: Dost Muhammad Khan

Several studies have proposed that vote tampering based on heuristic algorithms can manipulate voters' votes. It can be found from the analysis of the poll results of the 2016 US election that the frequency of "Trump won," which is generally considered a black swan phenomenon, is not low and even reached 16.8%. However, many models are unable to restore the generation of such a high frequency of black swan phenomena. In this study, the black swan phenomenon is successfully reproduced using a bias-generating agent-based election system model. By adjusting the tampering method, the frequency of the black swan phenomenon will change from 5% to 15%. From the simulation results, it can be observed that one of the possible causes of the black swan phenomenon is the tampering of the voting results, which leads to more biased voters, thus increasing the frequency of the winning elections. This study proposes that to obtain more realistic simulation results, it is necessary to introduce more realistic perceptual models for agents, rather than relying solely on random functions. Allowing agents to make mistakes for a reason should be an integral part of multi-agent-based simulation in the field of pairwise human simulation.

1. Introduction

Black swan phenomena frequently occur in electoral campaigns; for example, in the 2016 US election, the final election of Trump was considered as a black swan phenomenon in electoral campaigns [1]. An analysis of polling data related to the 2016 US election [2] shows that among polls conducted for the entire United States, the higher-trust "Google Consumer Surveys" results show that Trump was elected about 5% of the time in the poll results. Meanwhile, all polls across the United States show that Trump is elected with about 16.8% frequency. If every poll is considered a "real" election, then the frequency of black swans seems a bit high (even ignoring some of the less confident pollsters, Trump is still elected with a 10% frequency). Many scholars have proposed various voting models, and some of them have discussed whether it is feasible to use heuristic algorithms to tamper with voting results to achieve vote manipulation [3, 4]; other scholars have borrowed voting models to analyze the relationship between voters and social networks [5]; other scholars aim to identify a more just and democratic model of voting [6, 7].

However, few studies have used voting models to analyze the frequent occurrence of black swan phenomena. This study argues that this phenomenon occurs because the agents used in the voting model behave rationally and profitably. Hence, there are essentially no surprises and, therefore, no high-frequency black swans. In real life, voters cannot be rational all the time, and they may be blindly confident because of their own biases. Voters' overconfidence can have an impact on the voting results [3]. The purpose of this study is to introduce bias-generating voter models for voting systems to analyze how the high frequency of black swan phenomena is generated.

The purpose of this study is to apply it to a multi-agent model based on some models used to simulate perceptual behavior [8]. Since the perceptual behavior of individuals is inextricably linked to social phenomena, this study uses it to

see whether the simulation of perceptual behavior leads to more realistic simulation results; i.e., this study proposes an agent model that learns a false perception, called bias, from the environment. It is used to model the perceptual behavior of voters. To demonstrate its efficacy, it will be used to simulate the reproduction of Pavlov's dog experiment. This experiment is a basic example of bias due to the limitations of the received signal. Finally, it will be applied to a voting system to recover the high-frequency black swan phenomenon.

This study applies some of the techniques used to model the perceptual behavior of a single agent to a multi-agent system, enabling them to interact to analyze the impact of perceptual behavior on society as a whole. The significance of this study is not only to analyze the causes of the high-frequency black swan phenomenon in voting but also to make the following points. When simulating organisms that may behave emotionally, such as humans, it is necessary to introduce a function that can produce "irrational" (not just using a random function) behavior for the agent to obtain more realistic simulation results. At the same time, the boundaries of whether a behavior is "perceptual" are not clear, but since individual perceptual behavior is closely related to sociality, the more natural sociality observed in the multi-agent system can, in turn, support the demonstration of whether the simulation gets behavior "perceptual."

1.1. Research Purpose. This study argues that the high frequency of the black swan phenomenon in elections is caused by voter bias. That is, a few agents fall into cognitive bias due to more extreme environmental inputs, and their behavior leads to behavioral changes in other agents, ultimately leading to a runaway situation. To prove the point, this study will propose two different layers of models. One is the agent model, as an individual layer model for the whole simulation, which can simulate bias and generate bias from specific environmental inputs. The second model is the voting model, as a social layer model, which should be able to count the votes of the agents and perform some tampering with the results for the purpose of vote manipulation. This study argues that applying the general agent model to the voting model alone does not restore the emergence of the high-frequency black swan phenomenon, as its generation requires agent "overconfidence" [9]. Instead, when applying a bias-generating agent model, it is possible to model the overconfidence of some agents that generate the high-frequency black swan phenomenon.

Therefore, this study aimed to construct an agent model for generating bias and applying it to a simplified version of the voting model [3]. The result is that the simulation reproduces a high frequency of the black swan phenomenon and analyzes the possible causes of its generation.

2. Related Research Work

According to Nassim Nicholas Taleb, "one of the causes of the black swan phenomenon is overconfidence in experience" [9]. Knowledge acquired through observation and experience has significant limitations and vulnerabilities. This limited access to information can lead to bias and "overconfidence" in the results obtained. Therefore, in this study, bias is also related to the black swan phenomenon in elections. According to LA Suchman, all human behavior is based on improvisation in the current situation [10, 11]. Here, the current position consists of two parts, the brain's plan and the environment's input. It is in this "improvisation" that bias is generated, and the more extreme the setting [12, 13] and the more "crude" the plan [14], the more bias is caused. Therefore, in this study, the generation of bias and the brain's plan will be modeled separately.

A Wilczynski et al. proposed and proved that manipulating voting results using heuristic algorithms is feasible and effective in polynomial time [4]. In this study, a concurrent agent model is proposed. In this voting model, the intelligence gap that arises from the inability of the agent's social network to involve all agents is exploited for vote manipulation. The authors add a portion of protection to avoid uncontrollable outcomes so that the final simulation results in a controlled and valid voter vote manipulation. Thus, in a simplified version of this model, although the emergence of the black swan phenomenon can be seen, its frequency is low (≤3%), and it is also tricky to amplify this frequency by adjusting the parameters. Therefore, in this research work, the simple model with the protection mechanism removed will be improved to reproduce the high-frequency black swan phenomenon.

2.1. Agent-Based Simulation. Agent-based simulations, commonly used to study the relationship between individual behavior and overall change, are often used in studies related to voting simulation [3–5]. Agents will act according to a predetermined procedure so that interactions between agents will occur and eventually respond to overall changes [15–18]. As suggested by Argyris [35], to obtain more effective simulation results, some feedback needs to be introduced to allow the agents to learn, rather than just letting them "do what they are told." Some studies will empower agents with learning skills [20–22], based on reinforcement learning, where agents could learn from the environment and act on what they have learned. In this study, feedback learning is introduced. Still, it is also explored whether it can model the generation of bias when this feedback value is too large, even reaching a state of extreme overfitting. Neural networks were used as reinforcement learning models based on which the generation of bias was simulated.

3. Bias-Generating Agent

Here, in this section, a neural network-based bias-generating agent model is presented. This neural network enters the overfitting state faster by returning the same return value multiple times during the backtracking process. In this study, the overfitting state will be used to model the generation of bias. This principle will be applied to a simple dog model that reproduces Pavlov's dog experiments to demonstrate bias generation.

3.1. Neuron and Neural Network. This agent model is a neural network model, and each neuron contains the following elements:

$$\text{Neuron} = \langle I, O, \Omega, R_I, R_O, \text{AF}, \text{BT}\rangle, \tag{1}$$

where I is the set of inputs: $I = \{i_1, \ldots, i_n | i_x \in (0,1)\}$, where n is the number of inputs to the neuron, which is the output of other neurons or the initial input of the whole neural network, and if the neuron is in the input layer of the neural network, then we have $I = \{i_1, \ldots, i_n | i_x \in \{0,1\}\}$.

O is the output of the neuron: $O = \{o | o \in (0,1)\}$. Its value is the output of the activation function in the neuron. Ω is the weight of the input. $\Omega = \{\omega_1, \ldots, \omega_n | \omega_x \in (0,1)\}$, where n is the number of inputs to the neuron and each input I corresponds to a weight ω. The basic purpose of neural network backtracking is to update the weights. R_I is the set of return values received by this neuron. $R_I = \{r_{I_1}, \ldots, r_{I_n}\}$, where n is the number of neurons in the layer after this neuron. In the backtracking, each neuron passes a set of return values R_O to each neuron in its previous layer. $R_O = \{r_{O_1}, \ldots, r_{O_n}\}$. AF is the activation function: AF: $I \times \Omega \longrightarrow O$, and the sigmoid function is used as the activation function. BT is the backtracking function for updating the weights and calculating the passed return value. BT: $R_I \times \text{O} \longrightarrow \{\Omega, R_O\}$,

$$\omega_i = \omega_i - \text{LR} * \left(\sum_k^n r_{I_k}\right) \times o \times (1-o) \times i_i, \tag{2}$$

$$r_{O_i} = \left(\sum_k^n r_{I_k}\right) \times o \times (1-o) \times \omega_i, \tag{3}$$

where LR is the learning rate, $\text{LR} \in (0,1)$.

Figure 1 illustrates a neuron model with three inputs and three return values. These three inputs are weighted and summed by a sigmoid function to produce an output. This output is also involved in updating the weights and calculating the return value of the previous layer upon receipt of the return value.

A plural number of neurons can be constructed into a neural network with the following elements:

$$\text{Neuron Networks} = \langle I, O, \text{IL}, \text{HL}, \text{OL}, R\rangle, \tag{4}$$

where I is the set of inputs. $I = \{i_1, \ldots, i_n | i_x \in \{0,1\}\}$, where n is the number of inputs to the neural network. These inputs are also the inputs to the input layer. O is the set of outputs of the neural network. $O = \{o_1, \ldots, o_n | o_x \in (0,1)\}$, where n is the number of outputs of the neural network and each element has a value between 0 and 1. In the practical use of neural networks, this value needs to be defined in a normalized way. IL is the set of neurons in the input layer of the neural network. $\text{IL} = \{\text{neuron}_1, \ldots, \text{neuron}_n | \text{neuron}_x \text{Neuron}\}$, where n is the number of neurons in the input layer, i.e.,

$$|\text{IL}| = |I|, \tag{5}$$

where HL is the set of hidden layer neurons of a neural network. $\text{HL} = \{\text{neuron}_1, \ldots, \text{neuron}_n | \text{neuron}_x \text{Neuron}\}$, where n is the number of neurons in the hidden layer, and there is no limit to the number of neurons in the hidden layer. OL is the set of neurons in the output layer of the neural network. $\text{OL} = \{\text{neuron}_1, \ldots, \text{neuron}_n | \text{neuron}_x \varepsilon \text{Neuron}\}$, where n is the number of neurons in the output layer of the neural network, and the number is the same as the number of elements in the output set, i.e.,

$$|\text{OL}| = |O|, \tag{6}$$

where R is the set of returned values when the neural network is backtracked. $R = \{r_1, \ldots, r_n | o_x \in (0,1)\}$, where the value of n is the same as the number of elements of the output set, i.e.,

$$|R| = |O|. \tag{7}$$

Figure 2 shows a neural network with three inputs and one output, and the number of neurons in the hidden layer is 3.

3.2. Bias-Generating. This study exploits the overfitting property of neural networks to simulate the generation bias. Many scholars have been working to eliminate the effects of overfitting to achieve the best results for their programs [23, 24]. Others argue that benign overfitting is beneficial [25]. Schaffer C argues that it is the way the AI is used that determines whether overfitting is good or bad and that we cannot generalize about excess [26]. Sagi et al. [20] and Kirman [10] proposed an overfitting theory to explain perceptual learning and applied it to computer vision systems. In this study, the same return value was learned multiple times during backtracking learning of the neural network to reach the overfitting state quickly.

3.2.1. Operating Principle. The neural network shown in Figure 2 is used to illustrate the operating principle of this neural network. First, for convenience, define

$$\begin{aligned} \text{IL}_i &= \text{IL.neuron}_i, \\ \text{HL}_i &= \text{HL.neuron}_i, \\ \text{OL}_i &= \text{OL.neuron}_i, \end{aligned} \tag{8}$$

and then, for any $i_i \in I$, they are delivered to any input layer neuron IL_j as its input $\text{IL}_j.i_i$. For the neuron IL_i, it will calculate the output $\text{IL}_i.o$ and transmit it to the hidden layer neuron HL_j as its input $\text{HL}_j.i_i$. Similarly, for the neuron HL_i, it will calculate the output $\text{HL}_i.o$ and pass it to the output layer neuron OL_j as their input $\text{OL}_j.i_i$. Finally, the output $\text{OL}_i.o$ of the neuron OL_i in the output layer is the output o_i of the whole neural network. At that time, the neural network completes one output. Next, the case of backtracking is considered. The evaluation function $P(o)$ is used to evaluate the output,

$$P(o) = -\frac{1}{2} \times (d-o)^2, \tag{9}$$

where d is the ideal output. Then, if a backtracking operation is needed, for the output o_i, its return value r_i should be the derivative of $P(o_i)$, i.e.,

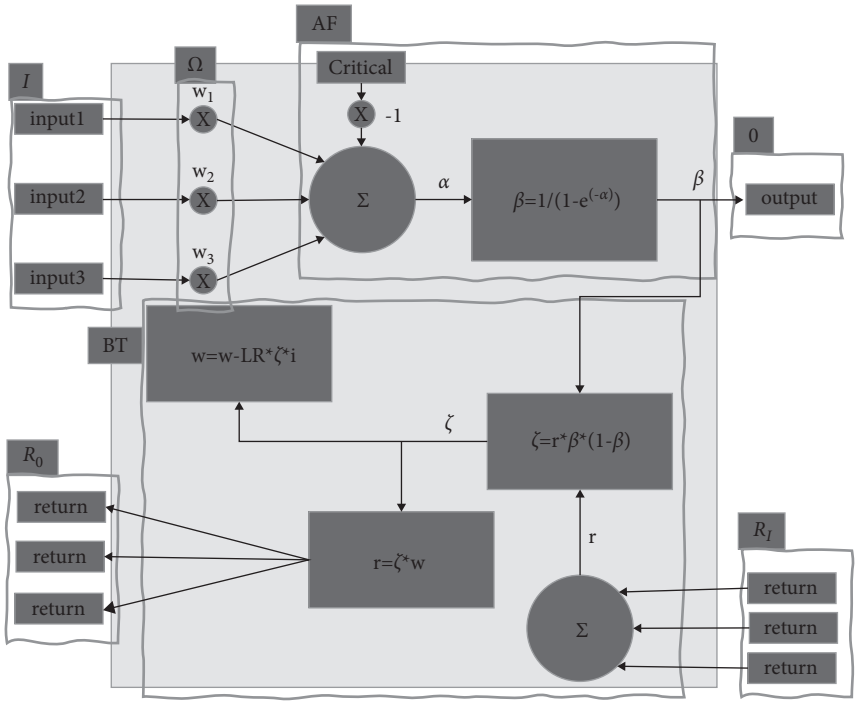

Figure 1: A 3-input, 3-return-value neuron model.

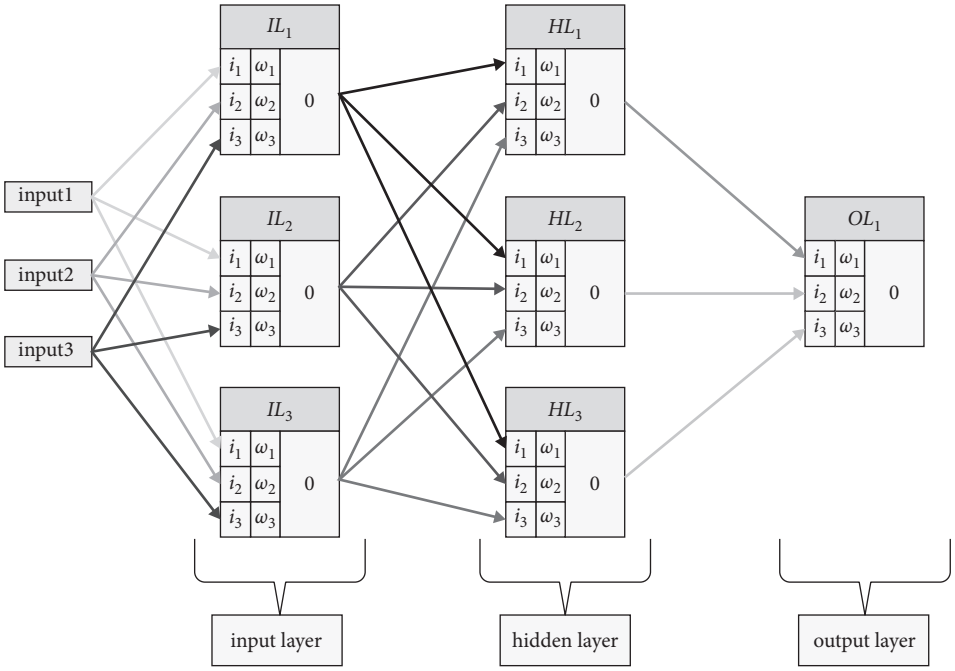

Figure 2: A 3-input, 1-output neural network model.

$$r_i = d_i - o_i. \tag{10}$$

Meanwhile, the return value r_i is also the return value of the output layer neuron OL_i, i.e.,

$$\mathrm{OL}_i.r_{I_1} = r_i. \tag{11}$$

It is worth noting that the output layer neurons all have only one return value. For any neuron OL_i, after getting the return value $\mathrm{OL}_i.r_I$, $\mathrm{OL}_i.\omega$ will be updated through (2), whereas the return value $\mathrm{OL}_i.r_{O_j}$ will be calculated through (3) and passed to the hidden layer neuron HL_j as the received return value $\mathrm{HL}_j.r_{I_i}$. For any neuron HL_j in the hidden layer, similarly, after getting the return value $\mathrm{HL}_i.r_I$, $\mathrm{HL}_i.\omega$ is updated with equation (2). Meanwhile, the return value $\mathrm{HL}_i.r_{O_j}$ passed to the input layer neuron IL_j is calculated by equation (3) and used as the received return value $\mathrm{IL}_j.r_{I_i}$. Finally, for the input layer neuron IL_i, after getting the return value $\mathrm{IL}_i.r_I$, $\mathrm{IL}_i.\omega$ is updated using equation (2). Currently, the whole neural network completes a backtracking operation. Based on the idea that overtraining leads to more frequent overfitting [27] if the model needs to simulate the generation of bias, it is sufficient to perform the above operation multiple times for the same return value r_i.

3.2.2. Model Features. As compared to other models of back propagation neural networks [28, 29], this model has two features:

(i) The individual neurons are more independent, and the output and backtracking of each neuron depend only on the input and return value it receives. This means that this neuron structure can be freely combined and is not limited to a specific framework.

(ii) In return value learning, simulating overtraining, the same return value is learned more than once for backtracking. This makes this neural network more prone to overfitting than other neural networks that only perform one return.

However, in turn, this makes the neural network somewhat disadvantageous compared with other BP neural networks.

(i) More independent neurons imply greater computational complexity, making this neural network much less computationally efficient than those neural network models that rely on matrices for backtracking computation. However, this drawback is quickly compensated on a vehicle that allows parallel computation.

(ii) A higher overfitting frequency often means that this neural network is substandard. Therefore, the number of backtracking learning needs to be controlled so that this number is not so high that it enters overfitting for arbitrary training data.

The first drawback is that this study's primary purpose is not to build a faster and more accurate neural network model. Instead, such a neuronal structure helps create an upgraded network to more conveniently increase or decrease the number of neurons in the input and output layers and the hidden layer.

This study is precisely designed to use overfitting to model perceptual behavior like bias for the second drawback. Hence, the higher frequency of overfitting is not a drawback in this study. Of course, it is not a good idea if the overfitting frequency is so high that it produces an overfitting state for any training dataset. Therefore, an excessive number of backtracking will not be used (usually 3 or 4 at most). The neural network only enters the overfitting state more quickly for the "more specific" training data sets.

3.3. Application of Bias-Generating Agent: Simulation of Pavlov's Dog. This section applies the bias-generating neural network introduced in the first three sections to a simple dog model to reproduce Pavlov's dog experiment to demonstrate bias generation.

3.3.1. Pavlov's Dog Experiment and Model. Pavlov's famous psychologist performed such an experiment with dogs [1, 30, 31], and every time a dog was shown a picture or a bell was rung before it was given something to eat. After some time, whenever the bell was rung, or the image was seen, the dog began to secrete saliva. Pavlov's dog experiments have a critical role in the innovation of one of the most important concepts in human thinking. While it happened quite an accident, Pavlov's famous experiments had a major impact on our understanding of how learning takes place and the development of the school of behavioral psychology [32].

This study considers Pavlov's dog as an excellent model to be used as an example for simulating bias generation.

For simulation convenience, the model of Pavlov's dog is defined as follows:

$$\mathrm{Dog} = <I, O, \mathrm{NN}, \mathrm{BR}>, \tag{12}$$

where I is the set of inputs. $I = \{i_f, i_m, i_p | i_f, i_m, i_p \in \{0, 1\}\}$, where i_f, i_m, i_p are whether to give food, bell, and picture, respectively, 0 is not given, and 1 is given.

O is the set of outputs. $O = \{o | o \in (0, 1)\}$. The magnitude of this value represents the probability that the dog will salivate (or the amount of saliva), and the closer to 1, the greater the probability (amount). This value depends on the output of the neural network and the brain, and this relationship is noted as a function $\mathrm{O}: \mathrm{NN} \times \mathrm{BR} \longrightarrow O$, where:

$$\mathrm{O}(\mathrm{NN}, \mathrm{BR}) = \begin{cases} \text{output}(\mathrm{NN}) \text{ where output } (\mathrm{BR}) = 0, \\ \text{output}(\mathrm{BR}) \text{ else}, \end{cases} \tag{13}$$

where NN is a neural network for dogs. $\mathrm{NN} \in$ Neuron Networks. Where the number of inputs to NN is 3, the number of hidden layer neurons is 3, and the number of outputs is 1; i.e., $|\mathrm{NN}.I| = 3, |\mathrm{NN.HL}| = 3, |\mathrm{NN}.O| = 1$.

According to equation (5)–(7), it can be derived that

$$\begin{aligned} |\mathrm{NN.IL}| &= 3, \\ |\mathrm{NN.OL}| &= 1, \\ |\mathrm{NN}.R| &= 1. \end{aligned} \tag{14}$$

BR is the brain of the dog, represented by the function br: $I \longrightarrow$ output (BR). There is:

$$br(\mathrm{I}) = \begin{cases} 0 \text{ where } i_f = 0, \\ 1 \text{ else.} \end{cases} \tag{15}$$

Figure 3 illustrates the structure and goal of the model, where the environment part is the input I, the reflexive nerve part consists of the neural network NN, and the brain part is the function $br(I)$. The goal of the model is to feed a specific amount of input I to the dog so that when $br(I) = 0$, O (NN, BR) = 1 for the dog.

3.3.2. Simulation Results. A model of a dog was built based on the models mentioned in the previous two subsections. First, the input signals are $I = \{1, 1, 0\}$ and $I = \{0, 0, 1\}$ and let the dog learn. During this period, the change in output O is observed if the input signal is $I = \{0, 1, 0\}$. The simulation results are shown in Figure 4. The simulated dog falls into a biased state after a certain number of inputs and drools even if it receives only the signal I = {0,1,0} (only music is heard).

Based on Pavlov's experiments with dogs, we know that prejudices produce results that can be changed, but over time they are reinforced and become "immutable," called "stereotypes." To prove that the model can also restore this situation, it is necessary to introduce some interference signals, as follows.

Based on the previous simulation, the interference signal $I' = \{0, 1, 0\}$ will be used as input from time to time for the simulation dog to learn. Again, the change in output O is observed at any moment if the input signal is $I = \{0, 1, 0\}$. The results are shown in Figure 5.

As can be seen in Figure 5, during the initial phase of training (within the first 40 moments), there is a significant change in the output O of the simulated dog due to the interference signal. Still, as time goes on, the interference signal becomes less influential on the results. This is consistent with the creation of "stereotypes" described earlier. In this section, a neural network-based bias generation agent model is proposed, which can be easily overfitted. It is expected to model the generation of biases and can eventually be fixed as "stereotypes." It was applied to a model of a dog to simulate Pavlov's dog experiments. Simulation results show that the model enables agents to make bias-like learning outcomes and that such learning outcomes are deepened and strengthened over time.

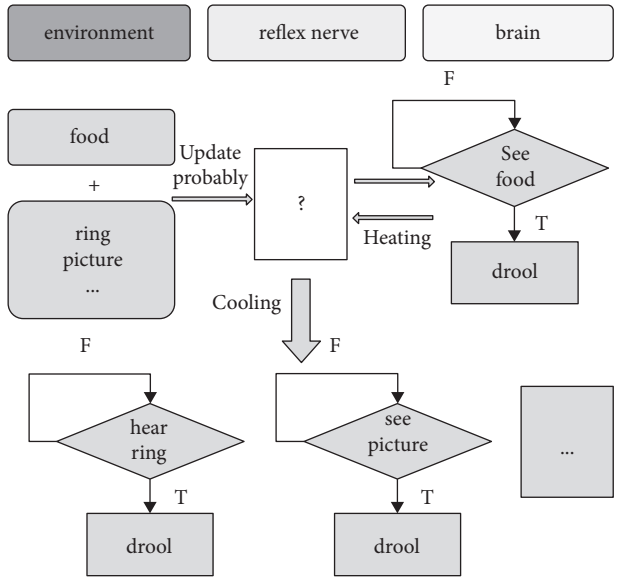

Figure 3: Pavlov's dog's model.

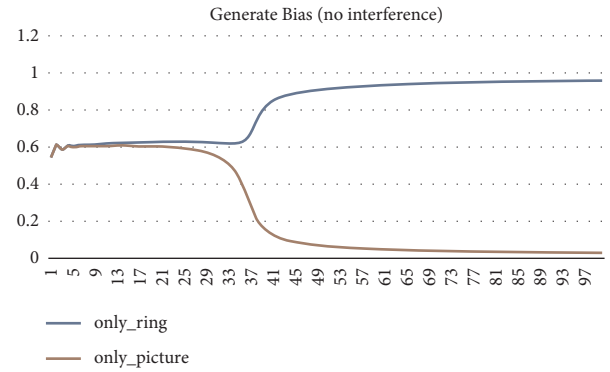

Figure 4: Simulation results of Pavlov's dog.

4. Voting Model with Polls

To apply biased generative agents to a voting system, it is first necessary to define a simple model of the voting system. The model used over here is a simplified version of the model proposed by A Wilczynski, which allows manipulation of voting results using heuristic algorithms [3]. Many cumbersome protections are ignored in the tampering operation, and the tampering is simplified to make the end of the tampering less "natural" so that the "black swan phenomenon" can be observed (even if its frequency is low).

4.1. Voting Model. The voting model is constructed as follows:

$$\text{Voting Model} = < V, C, \mathrm{PA}, \Gamma, \mathrm{Poll}, \mathrm{SN} >, \tag{16}$$

where V is the set of voters, $V = \{v_1, \ldots, v_n\}$, where n is the total number of voters. Each voter is composed of 6 more elements, which will be described in detail in Section 4.1.1. C is the set of candidates. $C = \{c_1, \ldots, c_m\}$, where m is the total number of candidates. This element is known for any voter v_i. PA is a poll function. PA: $V \times \mathrm{SN} \longrightarrow$ Poll is used to calculate the poll result Poll. This result can be either a simple count of voters' votes or a tampered poll result. The tampering method is mentioned in Section 4.1.2. Γ is the winning rule for this voting model in a general election. Γ: $V \times C \longrightarrow$ winner, where the winner $\in [1, m]$. The model uses a majority voting model, i.e., the candidate with the most votes wins. SN is the (average) density of the agent's

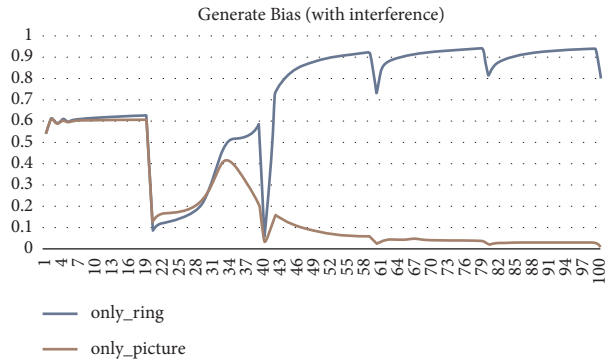

Figure 5: Simulation results of Pavlov's dog with interference.

social network, expressed as a distance, in the whole system. If the distance between a voter v_i and voter v_j is less than or equal to the SN, then v_i and v_j are called to be neighbors of each other.

4.1.1. Voter Model. For any voter v_i there is

$$v_i = <\succsim_i, \mathrm{TV}_i, \mathcal{N}_i, \mathscr{C}_i, \mathrm{T}_i, \mathscr{B}_i>. \tag{17}$$

The set $\succsim_i$ is the degree of voter v_i's preference for a candidate noted as $c_a \succsim_i c_b$ if v_i prefers candidate c_a over candidate c_b. The set $\succsim_i$ records the degree of voter preference for all candidates. TV_i is the set of votes for the candidate currently thought by voter v_i: $\mathrm{TV}_i = \{\mathrm{tv}_{i_1}, \ldots, \mathrm{tv}_{i_m}\}$. The value of TV_i is calculated by the thinking function T_i:

$$\mathrm{T}_i: \mathcal{N}_i \times \mathrm{Poll} \longrightarrow \mathrm{TV}_i. \tag{18}$$

The specific calculation method is to update Poll after subtracting the votes of the current neighbors $\mathcal{N}_i$ and their votes, and then, the changes in the votes of the neighbors are observed and watched and the value of TV_i in real time is updated. All voters default to their own unseen voters' votes without change. $\mathcal{N}_i$ is the neighbor of voter v_i; i.e., the set of all voters whose distance from voter v_i is less than or equal to SN: $\mathcal{N}_i = \{n_1, \ldots, n_k | n_1 \in V_{i^-} \times \mathrm{SN}\}$. $\mathscr{C}_i$ is v_i's confidence, representing voter v_i's perception of their voting value. Higher values of $\mathscr{C}_i$ indicate that voter v_i is more confident. If the value of $\mathscr{C}_i$ is set to positive infinity, it can be modeled that voter v_i is a loyal follower of the candidate $c_{\mathrm{top}(\succsim_i)}$, who ranks first in their preference ranking$\succsim_i$ (v_i will only vote for $c_{\mathrm{top}(\succsim_i)}$). $\mathscr{B}_i$ is the actual vote (ballot) of v_i; i.e., if a general election/vote were held at this time, v_i would vote for the candidate $\mathscr{B}_i$. The value of $\mathscr{B}_i$ is also calculated using the thinking function T_i, noted as follows:

$$\mathrm{T}_i: \mathrm{TV}_i \times \mathscr{C}_i \times \succsim_i \longrightarrow \mathscr{B}_i. \tag{19}$$

The specific behavior is as follows: v_i first their self-confidence $\mathscr{C}_i$ is added to each candidate, one by one based on TV_i, and if a candidate c_j has the highest number of votes after adding their self-confidence $\mathscr{C}_i$, then c_j is called to be the "probable winner (PW)." After confirming all PWs, candidates are selected from PWs for voting according to $\succsim_i$.

4.1.2. Tampering with Poll Results. First, the number of manipulable votes (MV) needs to be calculated based on the total number of voters $|V|$ and the average density SN of the social network. This value should not be so significant as to cause "suspicion" among voters [3]. Therefore, it is necessary to ensure that the minimum number of votes for each candidate is at least equal to the statistical sum of the votes of the neighbors $\mathcal{N}_i$, which can be observed by any voter v_i.

Wilczynski's method is to calculate $\mathcal{N}_i$ for each voter v_i in turn and then keep the maximum value. However, this method is a bit too computationally lengthy. For the convenience of subsequent simulations, the average density SN of the social network will be used in this model to simplify the calculation, so

$$\mathrm{MV} = |V| - \frac{\mathrm{SN} * (\mathrm{SN}+1)}{2}. \tag{20}$$

The purpose of tampering with the voting results is to try to manipulate voters to vote for the target candidate c_{target}. For this purpose, the method used here is to create an imaginary enemy, also known as the ghost candidate c_{ghost}, so that c_{target} and c_{ghost} are evenly matched. At this point, if $c_{\mathrm{target}} \succsim_i c_{\mathrm{ghost}}$ holds for most voters v_i, then the information gap (a single voter's social network cannot reach all voters) can be exploited to successfully canvass for c_{target}. In this study, the Borda count ranking [33, 34] will be considered as the level of preference of most voters; i.e., candidates with poor Borda count ranking will be regarded as ghost candidates c_{ghost}. For an "evenly matched" to hold, MVs need to be assigned to c_{target} and c_{ghost} so that they have more votes than the other candidates in the poll, and they end up with a similar number of votes or c_{target} is narrowly below c_{ghost} [3] and published as the Poll.

4.2. Results. Based on the above model, a model with 100 voters and 10 candidates was created for simulation.

4.2.1. Result 1—Whether to Tamper. First, whether the heuristic algorithm can manipulate the number of votes is observed. There are $c_{\mathrm{target}} = C.\mathrm{Borda\ Count}(6)$, $c_{\mathrm{ghost}} = C.\mathrm{Borda\ Count}(7)$, $\mathscr{C}_i = 3$, and $SN = 4$. The election frequencies (of 100 votes) of the target and ghost candidates were observed without tampering (flag_ tamper = 0) and with tampering (flag_ tamper = 1). The results are shown in Figure 6.

Two conclusions can be drawn from Figure 6:

(i) Heuristic algorithms can be used for voter vote manipulation purposes. (c_{target}'s election frequency increases from 0 to about 88%).

(ii) Heuristic algorithms can manipulate voters' votes, which may lead to the black swan phenomenon. (c_{ghost}'s election frequency also increases but is less than 3%, which is still acceptable as a category of the black swan phenomenon).

4.2.2. Result 2—Denser Social Networks. To observe the effect of the density of the social network on the impact of

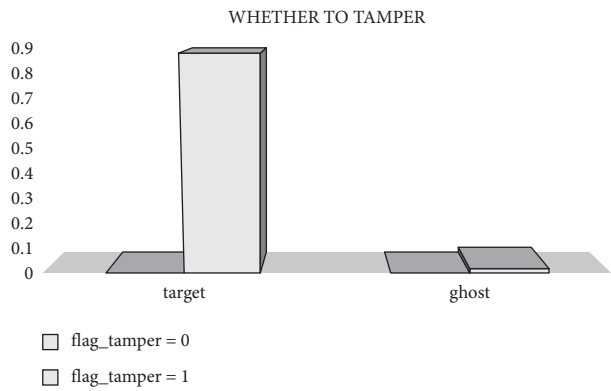

Figure 6: Result 1—whether to tamper.

manipulated voting, there are $c_{\text{target}} = C.\text{Borda Count}(1)$, $c_{\text{ghost}} = C.\text{Borda Count}(7)$, and $\mathscr{C}_i = 3$, and the election frequency of c_{target} at SN = 1 and SN = 6 is observed, respectively. The simulation results are shown in Figure 7.

It can be observed from the figure that the higher the density of social networks, the more difficult it is to manipulate voter voting using heuristic algorithms.

4.2.3. Result 3—Fanatic. The model can simulate the situation when the voters are fanatics. There are $c_{\text{target}} = C.\text{Borda Count}(1)$, $c_{\text{ghost}} = C.\text{Borda Count}(7)$, and SN = 1, and the election frequency of c_target when $\mathscr{C}_i = 3$ and $\mathscr{C}_i = 30$ is observed, respectively. The results are shown in Figure 8.

As can be seen from the graph, the higher the confidence level of the voters (the closer they are to fanatical believers), the more difficult it is to manipulate the voters' votes using heuristic algorithms.

4.2.4. Result 4—Wrong Ghost Candidate. Since it is almost impossible to predict the Borda count of a candidate, the tamperer needs to select c_{ghost} empirically. However, if the experience is biased and a candidate is more popular, thenc_{target} is selected as c_{ghost}, and they are bound to be "bad" consequences. To simulate this case, there are $c_{\text{target}} = C.\text{Borda Count}(7)$, $c_{\text{ghost}} = C.\text{Borda Count}(6)$, $\mathscr{C}_i = 3$, and SN = 1. The election frequencies of the target and ghost candidates were observed without tampering (flag_ tamper = 0) and with tampering (flag_ tamper = 1). The results are shown in Figure 9.

From Figure 9, we can see that when the popularity of c_{target} and c_{ghost} is misjudged, voters' votes will be absorbed more by c_{ghost}.

4.3. Summary of the Results. In this chapter, a simple model of a voting system with manipulated voting functionality is presented. Based on this model, the simulation simulates the results for four different cases. In terms of conclusions, there are five features:

(i) Heuristic algorithms can be used to manipulate voters' votes.

(ii) Heuristic algorithms may lead to the black swan phenomenon when manipulating voters' votes, but the frequency of the black swan phenomenon is still in a shallow range.

(iii) The higher the density of social networks, the more difficult it is to manipulate voters to vote.

(iv) The level of voter confidence has a significant impact on the outcome of the vote.

(v) Once voters' preferences for candidates are misjudged, rigging the vote through tampering is bound to backfire.

This suggests that the model can simulate the voting environment well, and these findings are consistent with the conclusions of A Wilczynski et al. Here, it can be seen that the black swan phenomenon occurs when someone tries to manipulate the voter's vote. Although it has a low probability of occurrence, this is a result that does not appear in A Wilczynski's model, including other models [35–37]. It is just that the frequency of such occurrences is still within the range of what could be recognized as a black swan phenomenon.

5. Bias-Generating Agent-Based Voting Model

In this study, a bias-generating agent model is proposed, which models the generation of bias. In Section 4, a simple voting model with a tampering mechanism is presented. In this section, these two models are combined and used to reproduce the high-frequency black swan phenomenon in elections. Agents that generate bias will act as individual layers, just like the agents in Section 4, and their behavior will have an impact on the overall social layer model (i.e., the voting model), which in turn will generate feedback that will allow agents to learn based on their changes.

5.1. BgAb-Voting Model. Similar to Section 4.1.1, this voting model is constructed in the following way:

$$BgAb\_\text{Voting Model} = < Bg\_V, C, \text{PA}, \Gamma, \text{Poll}, \text{SN} > . \quad (21)$$

Except for the set of voters Bg_V: $Bg\_V = \{\{bgv_1, \ldots, bgv_n\}$, the rest of the entire model is the same. Therefore, their descriptions are omitted. The difference between this model and the one in Chapter 4 lies mainly in the model of voter v_i.

5.2. Bias-Generating Voter. For any $bgv_i \in Bg\_V$, there is as follows:

$$bgv_i = < \succsim_i, \text{TV}_i, \mathscr{N}_i, \mathscr{C}_i, \text{T}_i, \mathscr{B}_i, \mathbb{N}_i > , \quad (22)$$

where$\succsim_i, \text{TV}_i, \mathscr{N}_i, \text{T}_i, \mathscr{B}_i$ are all defined in the same way as (17), so the descriptions are omitted. The main difference is that the confidence $\mathscr{C}_i$ is no longer a given value but is determined by the output of the neural network $\mathbb{N}_i \in \text{Neuron Networks}$, which learns Poll, $\mathscr{N}_i$, $\mathscr{B}_i$, and$\succsim_i$, denoted as $\mathbb{N}_i \times \text{Poll} \times \mathscr{N}_i \longrightarrow \mathscr{C}_i$. The following section will

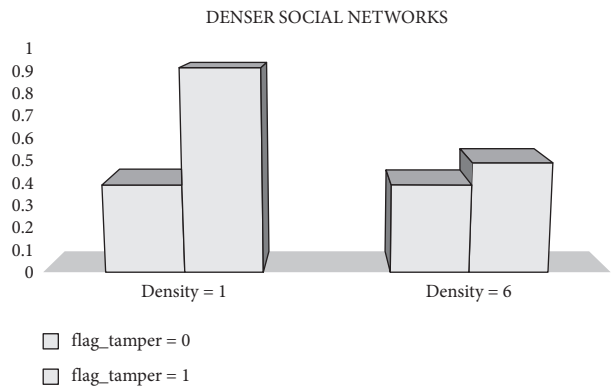

Figure 7: Result 2—denser social networks.

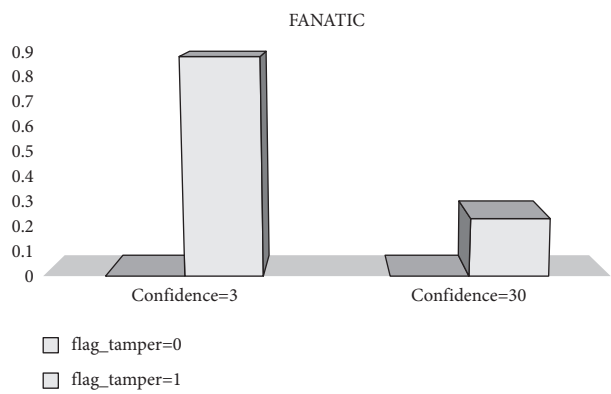

Figure 8: Result 3—fanatic.

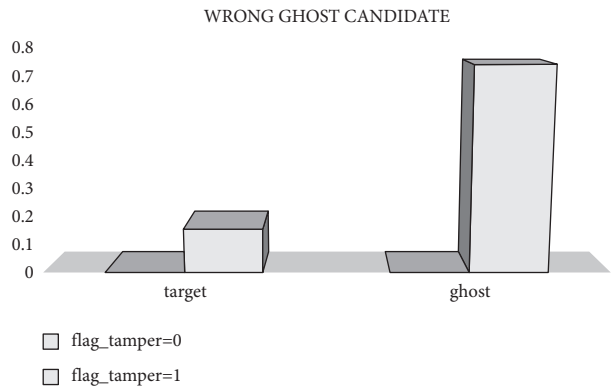

Figure 9: Result 4—wrong ghost candidate.

focus on the definition and parameter setting of this neural network.

5.2.1. Bias-Generating Voter's Neural Network. For $\mathbb{N}_i \in$ NeuronNetworks, it is defined as follows. The input $\mathbb{N}_i.I$ of this neural network has three elements corresponding to the answers of the three Boolean questions.

(i) if ($\mathscr{B}_i$ == Poll.winner). Was ballot the first-place winner in the poll?

(ii) *if* ($\succsim_i$.top == Poll.winner). Was the most popular candidate at the top of the poll?

(iii) if $\mathscr{N}_i$.top == Poll.winner. Was the first place in the neighborhood vote the first place in the poll?

The output of the neural network, $\mathbb{N}_i.O$, has only one element and is proportional to the degree of confidence: $\mathbb{N}_i.O \propto \mathscr{C}_i$. The number of different neuronal layers is as follows: $|\mathbb{N}_i.\mathrm{HL}| = |\mathbb{N}_i.\mathrm{IL}| = |\mathbb{N}_i.I| = 3$, and $|\mathbb{N}_i.\mathrm{OL}| = 1$. The return value $\mathbb{N}_i.R$ is proportional to the difference between the vote change values of $\mathscr{B}_i$ and $\mathscr{C}_i$ in Poll: $\mathbb{N}_i.R \propto \Delta(\mathrm{Poll}).\mathscr{B}_i - \mathscr{C}_i$.

5.2.2. Standardization. Since the output and return values of the neural network are between 0 and 1, normalization is necessary. The standardization is done as follows:

$$\mathscr{C}_i = \mathbb{N}_i.O * 10 + 1,$$
$$\mathbb{N}_i.R = \frac{\Delta(\mathrm{Poll}.\mathscr{B}_i) - \mathscr{C}_i}{10}. \tag{23}$$

Based on the principle of "one person, one vote," in this study, bias is considered to be generated when the confidence level of voters is high. In the subsequent analysis of the results, the relationship between the confidence level of those voters whose preferred candidate is the phantom candidate and the average confidence level of all voters will be considered to analyze the relationship between bias generation and the black swan phenomenon.

5.3. Results. Based on the above model, similar to Section 4.2, a model with 100 voters and 10 candidates is created for simulation. There are $c_{\mathrm{target}} = C.\mathrm{Borda\,Count}(6)$, $c_{\mathrm{ghost}} = C.\mathrm{Borda\,Count}(7)$, and $\mathrm{SN} = 4$. The election frequencies of the target and ghost candidates were observed without tampering (flag_ tamper = 0) and with tampering (flag_ tamper = 1). The results are shown in Figure 10.

Comparing Figure 6, it can be seen that the frequency of c_{ghost} elections has increased significantly (about 9%), a figure that is already very close to the frequency of the high-frequency black swan phenomenon mentioned in Chapter 1, which is about 10%. Adjusting different parameters still gives similar results, with c_{ghost} being elected between 5% and 15%. Figure 11 illustrates a portion of these results of whether to temper or not.

Continuing to analyze the experimental data in Figure 10, it is obtained that the average confidence of voters under this simulation is about 5.7; i.e., $\overline{\mathscr{C}} = 5.7$. There are $c_{\mathrm{target}} = C.\mathrm{Borda\,Count}(6)$, $c_{\mathrm{ghost}} = C.\mathrm{Borda\,Count}(7)$, and $SN = 4$. The election frequency of c_{ghost} is observed when $\mathscr{C}_i \equiv 6$ and $\mathscr{C}_i = \mathrm{std}(\mathbb{N}_i.O)$. The results are shown in Figure 12.

As can be seen in Figure 12, even at the same confidence level, whether the voter is bias-generating or not still has a significant impact on the emergence of the high-frequency black swan phenomenon. To further analyze the relationship between the generation of bias and the black swan phenomenon, the analysis continues with (Figure 10) the analysis of the percentage of voters who identify c_{ghost} as their preferred candidate ($\succsim_i$.top = c_{ghost}) with an average

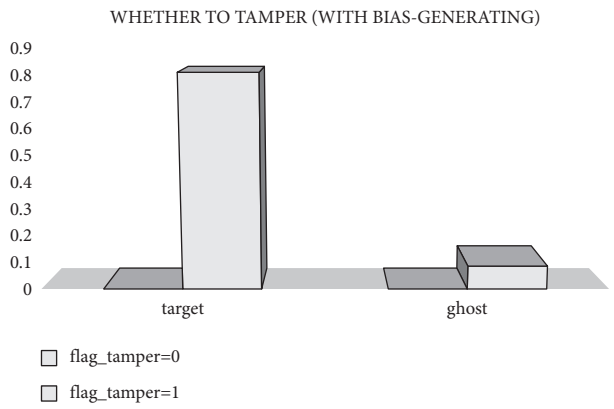

Figure 10: Result—whether to tamper (with bias-generating).

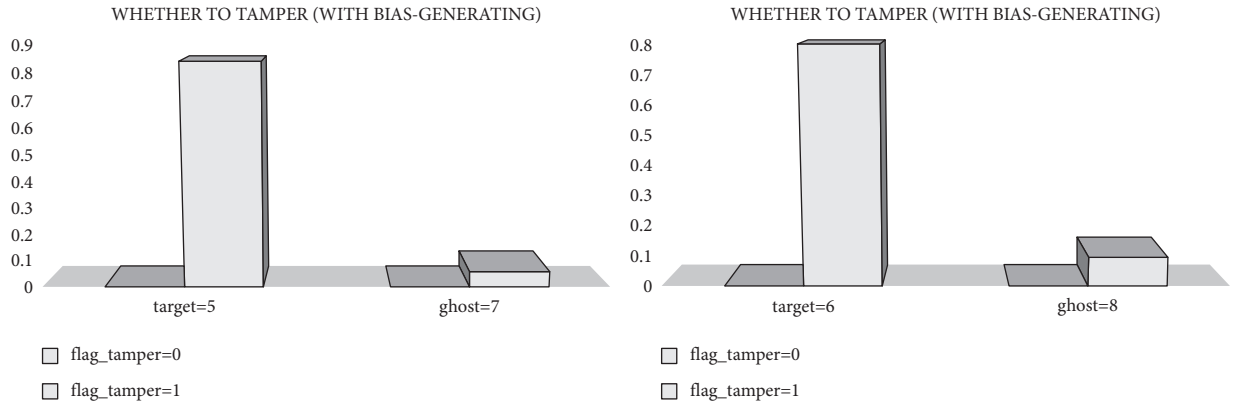

Figure 11: More result—whether to tamper (with bias-generating).

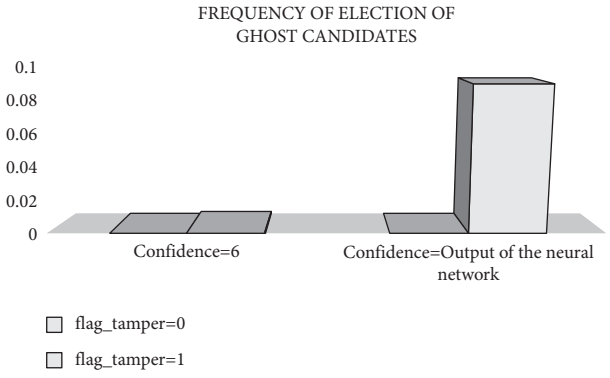

Figure 12: More result—frequency of election of ghost candidates.

confidence level higher than the average confidence level of all voters in the entire model in the following three scenarios.

(i) In any case.

(ii) Only if c_{ghost} is elected.

(iii) Only if $\overline{c_{\text{ghost}}}$ is elected.

The results are shown in Figure 13.

As shown in Figure 13, in the case that c_{ghost} is elected, voters who have c_{ghost} as their preferred candidate will have a higher confidence level than the overall average confidence level. On the contrary, in the case that c_{ghost} is not elected, their confidence level is not much different from the normal situation.

The bias-generating agent model introduced in the earlier section is combined with the voting model presented in Chapter 4 and simulates the generation of high-frequency black swan phenomena. The simulation results show that black swans appear more frequently in the model with bias-generating voters than in the model without bias-generating voters and are closer to the frequencies mentioned in the earlier section. At the same time, it can be seen that whether bias is created or not, there is a prerequisite for the black swan phenomenon to occur. That is, the results of public opinion polls are tampered with. This suggests that one of the possible causes of the black swan phenomenon is that the results have been tampered with to manipulate the vote. The reason for the increased frequency of the black swan phenomenon is that some voters have been misled by the results of such manipulated polls and have thus generated bias. In this study, it is considered possible that some voters who like c_{ghost} are more inclined to vote for c_{ghost} because they develop a bias and thus have a higher level of trust, and if these voters happen to be closer to each other, they will drive their neighbors to vote for c_{ghost} as well because of their social network, which eventually leads to

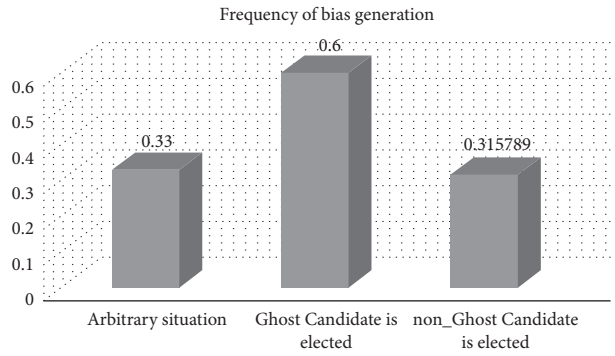

Figure 13: More result—frequency of bias generation.

the frequent occurrence of the black swan phenomenon. This is also corroborated in Figure 4.2.3, where a higher degree of confidence can make the tampering results uncontrollable.

6. Conclusion

This research work simulates the high frequency of the black swan phenomenon, which demonstrates that using an agent model that generates perceived behavior (e.g., bias) in a multi-agent model can simulate and reproduce more natural social phenomena. It is argued that the high frequency of the black swan phenomenon in elections is caused by voter bias. That is, a few agents fall into cognitive bias due to more extreme environmental inputs, and their behavior leads to behavioral changes in other agents, ultimately leading to a runaway situation. To demonstrate this, a bias-generating agent model is first proposed that enables the agent to learn some misinformation from the input of the environment, and a simulation of Pavlov's dog is used to demonstrate that it can provide a model of bias generation. Then, a simple voting model with polls is proposed, and after experimental simulations, it is shown that it can achieve the purpose of controlling voters' votes through heuristic algorithms. It is also found that the black swan phenomenon sometimes occurs when tampering with voting results. To reproduce the high frequency of the black swan phenomenon, these two models were combined so that voters would be biased and "overconfident" due to some misinformation. This directly leads to an increase in the frequency of the black swan phenomenon, reaching values similar to the actual data. Analysis suggests that one of the possible causes of the black swan phenomenon is that the poll results were tampered with, which led to more bias among voters, thus deepening the frequency of the black swan phenomenon. Compared with the studies by A Wilczynski et al. [3–5], who utilized several voter models to simulate the voting proceeding and finally to achieve simulated voting results in various scenarios. The present research work focuses more on the learning behavior of the voter model in the voting system by adding a learning function to the voter model. Compared with the studies by Alan Kirman et al. [10–37], who introduced a learning capability for the agent that allowed the agent to learn from the environment to make more correct decisions and used stochastic functions (e.g., epsilon-greedy method) to simulate "unexpected" situations. This study also introduces perceptual learning mechanisms for the agents in agent-based simulations, which allow the agents to perform some "nonoptimal solution" behaviors without relying on random functions. The advantage of this is that it allows agents to make mistakes "with reason," and "unexpected" situations can be explained. These bias the voters in the voting model and ultimately lead to the frequent occurrence of the black swan phenomenon. This study is not only intended to identify the reasons for the frequent occurrence of the black swan phenomenon. It also aims to show that when simulating humans or other organisms considered to have perceptual minds, it is necessary to introduce more realistic perceptual models for agents rather than to rely on random functions, to obtain simulation results that are closer to reality. Allowing agents to make mistakes "for a reason" should be an integral part of multi-agent-based simulations in the field of human simulation. This bias-generating agent model can be extended to other scenarios to support the findings of this study.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors have no competing interest.

References

[1] J Jerit and J Barabas, "Partisan perceptual bias and the information environment," *The Journal of Politics*, vol. 74, no. 3, pp. 672–684, 2012.

[2] A. Wiener, "Trump and the end of taken for grantedness: when the exception becomes the rule," in *Proceedings of the Norms Research in International Relations (IR) Theory*, Cambridge, UK, November 2006.

[3] Kaggle, "Election Polls [Dataset]," 2016, https://www.kaggle.com/fivethirtyeight/2016-election-polls.

[4] A. Wilczynski, "Poll-confident voters in iterative voting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 2205–2212, 2019.

[5] D Baumeister, A. K Selker, and A Wilczynski, "Manipulation of Opinion Polls to Influence Iterative Elections," in *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, Auckland New Zealand, May 2020.

[6] A Wilczynski, *Interaction Among Agents via a Social Network in Computational Social choice*, Université Paris sciences et lettres, Paris , France, 2018.

[7] J. D. Huber, "The vote of confidence in parliamentary democracies," *American Political Science Review*, vol. 90, no. 2, pp. 269–282, 1996.

[8] L Suchman and L. A Suchman, *Human-machine Reconfigurations: Plans and Situated actions*, Cambridge University Press, UK, 2007.

[9] T Tully, "Pavlov's dogs," *Current Biology*, vol. 13, no. 4, pp. R117–R119, 2003.

[10] A. Kirman, "Learning in agent-based models," *Eastern Economic Journal*, vol. 37, no. 1, pp. 20–27, 2011.

[11] T Kobayashi, S Takahashi, and M Kunigami, "Is there innovation or deviation? Analyzing Emergent Organizational Behaviors through an Agent Based Model and a Case design," in *Proceedings of the 5th International Conference on Information, Process, and Knowledge Management (eKNOW 2013)*, pp. 166–171, Nice, France, March 2013.

[12] D. M. Hawkins, "The problem of overfitting," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[13] N Srivastava, G Hinton, and A Krizhevsky, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[14] P. L. Bartlett, P. M. Long, and G. Lugosi, "Benign overfitting in linear regression," *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30063–30070, 2020.

[15] C Schaffer, "Overfitting avoidance as bias," *Machine Learning*, vol. 10, no. 2, pp. 153–178, 1993.

[16] I. V Tetko, D. J Livingstone, and A. I Luik, "Neural network studies. 1. Comparison of overfitting and overtraining," *Journal of Chemical Information and Computer Sciences*, vol. 35, no. 5, pp. 826–833, 1995.

[17] W Jin, Z. J Li, and L. S Wei, "The improvements of BP neural network learning algorithm,"vol. 3, pp. 1647–1649, in *Proceedings of the WCC 2000-ICSP 2000. 2000 5th international conference on signal processing proceedings. 16th world computer congress 2000*, vol. 3, pp. 1647–1649, IEEE, Beijing, China, August 2000.

[18] S Ding, C Su, and J Yu, "An optimizing BP neural network algorithm based on genetic algorithm," *Artificial intelligence review*, vol. 36, no. 2, pp. 153–162, 2011.

[19] C Argyris, "Single-loop and double-loop models in research on decision making," *Administrative Science Quarterly*, vol. 21, pp. 363–375, 1976.

[20] M Ziegler, R Soni, and T Patelczyk, "n electronic version of Pavlov's dog," *Advanced Functional Materials*, vol. 22, no. 13, pp. 2744–2749, 2012.

[21] S. A McLeod, *Pavlov's dogs*, Simply Psychology, Michigan, 49418, USA, 2007.

[22] J. L. García-Lapresta and M Martínez-Panero, "Borda count versus approval voting: a fuzzy approach," *Public Choice*, vol. 112, no. 1, pp. 167–184, 2002.

[23] P Buisseret, *A Political Economy of the Separation of Electoral origin*, University of Warwick, UK, 2013.

[24] D. Sagi, "Perceptual learning in vision research[J]," *Vision Research*, vol. 51, no. 13, pp. 1552–1566, 2011.

[25] N. N Taleb, *The Black Swan: The Impact of the Highly improbable*, Random House, NY, USA, 2007.

[26] L. A. Suchman, *Plans and Situated Actions: The Problem of Human-Machine communication*, Cambridge University Press, UK, 1987.

[27] R. L Calvert, "Robustness of the multidimensional voting model: candidate motivations, uncertainty, and convergence," *American Journal of Political Science*, vol. 29, pp. 69–95, 1985.

[28] Y Bachrach, E Elkind, and P Faliszewski, "Coalitional voting manipulation: a game-theoretic perspective," *IJCAI*, vol. 10, pp. 978–981, 2011.

[29] A Bassi, "Voting systems and strategic manipulation: an experimental study," *Journal of Theoretical Politics*, vol. 27, no. 1, pp. 58–85, 2015.

[30] D Schultz, C. E Izard, and B. P Ackerman, "Children's anger attribution bias: relations to family environment and social adjustment," *Social Development*, vol. 9, no. 3, pp. 284–301, 2000.

[31] R. A. Boakes, "The impact of Pavlov on the psychology of learning in English-speaking countries," *The Spanish Journal of Psychology*, vol. 6, no. 2, pp. 93–98, 2003.

[32] H. Nasrazadani and M. Mahsuli, "Probabilistic framework for evaluating community resilience: integration of risk models and agent-based simulation," *Journal of Structural Engineering*, vol. 146, no. 11, 2020.

[33] Yi Hwang, "Visualized Co-simulation of adaptive human behavior and dynamic building performance: an agent-based model (ABM) and artificial intelligence (A.I.) approach for smart architectural design," *Sustainability*, vol. 12, no. 16, 2020.

[34] H. Tatapudi, R. Das, and T. K. Das, "Impact Assessment of Full and Partial Stay-At-Home Orders, Face Mask Usage, and Contact Tracing: An Agent-Based Simulation Study of COVID-19 for an Urban Region," *Medical Letter on the CDC & FDA*, vol. 2, 2020.

[35] "Sociology - artificial societies and social simulation; investigators at technical university of crete report findings in artificial societies and social simulation (an agent-based model for simulating inter-settlement trade In Past Societies)," Politics & Government Business, 2020.

[36] J. Kirshner, J. Hardy, S. Wilczynski, and J. E. Shively, "Cell–cell adhesion molecule CEACAM1 is expressed in normal breast and milk and associates with β1 integrin in a 3D model of morphogenesis," *Journal of Molecular Histology*, vol. 35, no. 3, pp. 287–299, 2004.

[37] P. Emerson, "The original Borda count and partial voting," *Social Choice and Welfare*, vol. 40, no. 2, pp. 353–358, 2013.