Proceedings of CONF-CDS 2025 Symposium: Data Visualization Methods for Evaluation
DOI: 10.54254/2755-2721/2025.PO25141

# *Behavior-determining AI in Computer Games: A Literature Review*

**Xuancheng Lu[1*], Junlin Zhang[2], Yixuan Chai[3], Louis Ling[4]**

[1]*Shanghai Starriver Bilingual School, Shanghai, China*
[2]*Cogdel Cranleigh High School, Wuhan, China*
[3]*Haidian Foreign Language School, Beijing, China*
[4]*Dao Xiang Hu School, Beijing, China*
**Corresponding Author. Email: 2231988446@qq.com*

***Abstract.*** We studied different kinds of algorithm used in games that improve the performance of the AIs, including the frequency-adjusted Q-Learning (FAQ Learning), the robust multi-agent Q-Learning (RoM-Q Learning), and deep recurrent Q-Learning (DRQN). We reviewed a paper for each one of the algorithms and create this review paper presenting the use of different Q-learning algorithms on AI to improve their performance in games. Through improving the performing of AI in these areas, we are able to find something attractive in the different kind of behavior the AI does which is stunning and surprising. So, we can have a more intelligent game systems and become more real.

***Keywords:*** Robust Multi-agent Q-learning, Q-learning, Frequency Adjusted Q-learning, Deep Recurrent Q-Learning, Minimax Q- Learning

## 1. Introduction

In the realm of artificial intelligence and machine learning, the applications of advanced algorithms in computer games have gained significant attention. Games, with their dynamic and uncertain environments, present unique challenges for artificial intelligence (AI) systems. Different game designers have their own ways to adjust the AI to do different behaviors in different games, which is what the players want to play with. To meet these challenges, researchers have explored various Q-learning algorithms tailored to improve AI performances in games. This study reviews three key topics related to the use of Q-learning algorithms in AI for games: Frequency Adjusted Q-Learning (FAQ Learning), Robust Multi-agent Q-Learning (RoM-Q Learning), and Deep Recurrent Q-Learning (DRQN). FAQ Learning addresses the complexities of multi-agent learning environments, where the behavior of individual agents is influenced by the actions of others. By aligning with predictions derived from evolutionary models, FAQ Learning aims to enhance the robustness and efficiency of multi-agent learning systems. On the other hand, RoM-Q Learning focuses on developing strategies that can defend against adversarial attacks in cooperative games. This algorithm incorporates a temporal difference learning framework to find policies that are resilient to such attacks. In the end, DRQN modifies the traditional Q-learning model to train AI for playing

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.PO25141

First- Person-Shooting (FPS) games, demonstrating impressive performance in both full and limited death matches.

## 2. Background

With the development of the game industry, players have now become picky about the artificial intelligence in games, which is indeed a major factor in all kinds of games - whether the non-player characters (NPCs) or the AI enemies. Thus, the performance of the agents in the games is very important for the experience of players and the selling of the games.

Agents in the past often decided their actions by simple behavior trees or fixed actions that were decided by the game developers. However, these actions are often predictable, so game researchers tried to develop many other algorithms that allow the agents to decide the actions themselves and thus improve the game experience by making the agent actions unpredictable.

### 2.1. Q-Learning

Q-Learning is an algorithm that makes the agent learn a Q function that represents the cumulative rewards from past actions. It modifies the temporal difference algorithm and uses it to create the updating function.

$$Q\left(s,a\right)\leftarrow Q\left(s,a\right)+\alpha\left[r+\gamma\underset{a'}{\max}Q\left(s',a'\right)-Q\left(s,a\right)\right] \tag{1}$$

After taking action an in states, the agent observes the reward r and the next states′. The target value $r+\gamma V^{T}(s')$ is computed. The value $V^{T}(s')$ is calculated using the function introduced in the previous paragraph. The temporal difference error $r+\gamma V^{T}(s')-Q(s,a)$ represents the difference between the target value and the current Q-value, and eventually the Q- value is updated by multiplying α, a fraction of the temporal difference error to the current Q-value.

However, Q-Learning often performs poorly in any condition that is variable and unknown, since it can only find an optimal solution for a specific, fixed environment. Thus, more algorithms have been developed based on Q-Learning to fit in different environments, such as Minimax-Q learning.

### 2.2. Minimax Q-Learning

Minimax Q-Learning improves the way to calculate the Q- values by trying to choose the highest reward while keeping the opponent's reward the least.

$$V\left(s'\right)=\underset{\pi_1}{\max}\underset{\pi_2}{\min}\sum_{a_1,a_2}Q\left(s',a_1,a_2\right)\pi_1\left(a_1\middle|s'\right)\pi_2\left(a_2\middle|s'\right) \tag{2}$$

key differences between standard Q-learning and FAQ learning approaches.

First, Q-learning aims to maximize discounted benefits in multi-state environments, while the paper focuses on single- state multi-agent Q-learning. The method iteratively updates Q-values using a Boltzmann exploration strategy to balances exploration and exploitation. FAQ learning primarily correct

The $V(s')$ here is the calculated value that will be updated to the model. Minimax Q-Learning assumes that the loss of the opponents is equal to the reward of the agent, and it will go through all

DOI: 10.54254/2755-2721/2025.PO25141

possible combinations of opponents' actions and allows the algorithm to find the action that minimizes the opponents' rewards.

After researchers improve the algorithms, they then question: What if there are multiple opponents? To address this, developers continue researching and developing, eventually Q-learning's evolutionary trajectory, achieving better alignment with evolutionary model assumptions under continuous time limits.

The FAQ learning update function is given by:

$$Q_i\left(t+1\right) \leftarrow Q_i\left(t\right) + x_i\alpha\left(r_i\left(t\right) + \gamma \underset{j}{\text{argmax}} Q_j\left(t\right) - Q_i\left(t\right)\right) \tag{3}$$

In contrast, standard Q-learning uses:

$$Q_i\left(t+1\right) \leftarrow Q_i\left(t\right) + \alpha\left(r_i\left(t\right) + \gamma \underset{j}{\text{argmax}} Q_j\left(t\right) - Q_i\left(t\right)\right) \tag{4}$$

creating many distinct algorithms. Interested in understanding how these algorithms work and evaluating their performance under different conditions, we investigated several notable approaches. From these, we selected three: Frequency-Adjusted Multi-agent Q-learning, Robust Multi-agent Q-Learning, and Deep Reinforcement Learning. We analyzed their respective papers, and uncovered the logic behind their formulations.

## 3. Frequency adjusted multi-agent Q-learning

The combination of the thinking theory framework in NPC behavior logic and Q-learning was explored by [1] in their work on Frequency Adjusted Multi-agent Q-learning. The authors compared the advantages and disadvantages between multi-agent learning and single agent learning, proposing a new Q-learning method called Frequency Adjusted Q- Learning (FAQ Learning).

Multi-agent learning is more complex than single-agent learning and lacks solid theoretical support. Recent developments have combined evolutionary game theory with reinforcement learning, offering new perspectives for multi- agent learning. This includes establishing connections between Q-learning dynamics and replicator dynamics in evolutionary game theory. The replicator dynamics formally define population changes over time, where a population consists of individuals belonging to species that correspond to pure actions available to a learner.

The utility function $r_i(t)$ assigns rewards to actions per- formed, analogous to Darwinian fitness for each species i. The distribution of individuals across different strategies can be described by a probability vector equivalent to a player's policy. Evolutionary pressure through natural selection is modeled by replicator equations, where successful strategies with above-average payoffs grow while less successful one's decay. However, empirical findings demonstrate that actual Q- learning behavior deviates from evolutionary game theory predictions, with models often presenting more idealized out- comes than observed in practice. The relationship between the learning rate α and Q-learning proves crucial for achieving optimal convergence speed and quality. The paper highlights

To evaluate the relationship between FAQ and Q-learning, the authors conducted experiments using three game theoretic scenarios:

- The prisoner's dilemma (single pure Nash equilibrium)
- Battle of the sexes (two pure Nash equilibrium and one mixed strategy equilibrium)
- Matching pennies (single mixed strategy Nash equilibrium)

DOI: 10.54254/2755-2721/2025.PO25141

These experimental games were selected to comprehensively test FAQ learning's performance across different equilibrium types. The results suggest that FAQ learning offers improved theoretical grounding and practical performance compared to standard Q-learning in multi-agent settings.

## 4. Robust multi-agent Q-learning in cooperative games with adversaries

[2] introduces an algorithm the authors developed, robust multi-agent Q-learning (RoM-Q,) and compared the algorithm with other algorithms under the environment of multi-agent systems (MAS). The authors are inspired by real world networks and hopes to allow RoM-Q plays with the best performance in competitive games.

To begin with, the RoM-Q is a Q-learning method that aim for MAS, its target is to find a robust solution to defend attackers' actions. The RoM-Q is based on temporal difference learning, which uses the difference between the recent expectation and the future predict to modify the value functions, in order to find the best solution without the environmental models. However, there are also differences between RoM-Q and temporal difference learning. While updating Q values, RoM-Q will assume there are K attackers and traverse all possible combinations of attacks, and choose the action that has the least Q-value to break the attack.

The basic logic of the algorithm modifies is shown by a pseudocode and a function in the paper, and we will explain it here. Firstly, after initializing the inputted Agent set N, state space S, action space A, learning rate α, discount factor γ, exploration rate ϵ, attack size κ, and problem horizon h, the program will make its decision based on the current state S. It will follow the ϵ-greedy policy, which is the algorithm that keeps the balance between the newly explored data and the existing data. Then, it will execute the choice and observe the reward r and the next state S. If S is in terminal stage, the system will rest, otherwise it will update the Q-value by using the algorithm, going though all combinations, solve linear equations and obtain the decision, then repeat.

The function looks like this:

$$V^T\left(S\right) = \min_{j\in\mathscr{C}_K}\min_{a_j}\max_{\pi_{-j}}\sum_{a_{-j}} Q\left(S, a_{-j}, a_j\right)\pi_{-j}\left(a_{-j}\middle|S\right) \tag{5}$$

where $\min_{j\in\mathscr{C}_K}$ iterates over all possible combinations of κ adversaries, $\min_a$j selects the action $a_j$ that minimizes the Q-value needed while facing each possible attack j, $\max_\pi$–j is the best action taken while the agent is not under attack, and a–j Q(S, $a_{-j}$, $a_j$)$\pi_{-j}$($a_{-j}$|S) computes the expected Q-value in state S.

For the updating of Q-value, the algorithm uses the same way of updating with Q-learning, which has been mentioned in the background:

$$Q\left(s,a\right) \leftarrow Q\left(s,a\right) + \alpha\left(r + \gamma V^T\left(s'\right) - Q\left(s,a\right)\right) \tag{6}$$

To examine the effectiveness of RoM-Q learning, the com- parison between RoM-Q learning and other algorithms is necessary. The researchers decided to compare RoM-Q learning with Q-learning and minimax-Q learning. The experiment is set in a network with two nodes, and here are the variables that appear in the experiment:

DOI: 10.54254/2755-2721/2025.PO25141

Table 1. Symbols

| Defense Algorithm | δ = 0.1 | δ = 0.4 | δ = 0.8 |
|---|---|---|---|
| Q-learning | 5.6 | 3.3 | 2.5 |
| minimax-Q | 6.2 | 4.9 | 4.5 |
| RoM-Q | 6.8 | 7.2 | 5.2 |

The algorithms take these parameters in both attack-existing environments and no-attack environments. For the no-attack condition, the three algorithms perform similar, with RoM- Q in the middle, minimax-Q at the top and Q-learning at the bottom. This data shows that RoM-Q is always in the safe zone, and though minimax-Q achieves the high reward in smaller number of actions, its robustness is not as large as RoM-Q. In the second experiment, the three algorithms take turns in being attackers, and then their response under attacks are being tested. Under each attack algorithms, when the probability of attack δ increases, all systems will have lower performance. Here are the estimated values when δ is about 0.1, 0.4, and 0.8.

Table 2. Test performance under q-learning-based attacks

| Variable | Meaning |
|---|---|
| $c_i$ | The limit of tasks a node can hold. |
| $rarr_i$ | The probability of a task being given to the node. |
| $aexec_i$ | complete a task, reducing the task count by 1. |
| $pexec_i$ | Cost of the action |
| $aoff_i$ | transfers one task to another node. |
| $poff_{i,j}$ | cost of transferring. |
| $prepr_i$ | punishment of having number of tasks more than $c_i$. |
| $u_i$ | Survival reward |

DOI: 10.54254/2755-2721/2025.PO25141

Table 3. Test performance under minimax-q-based attacks

| Parameters | Value | | |
|---|---|---|---|
| Training samples $S_{train}$ | 1,000,000 | | |
| Evaluation samples $S_{eval}$ | 20,000 | | |
| Trials I | 40 | | |
| Learning rate α | 0.01 | | |
| Exploration rate ϵ | 0.1 | | |
| Discount factor γ | 0.9 | | |
| Reward u | [8, 8] | | |
| Over-flow penalty $p^{over}$ | [100, 100] | | |
| Capacity c | [3, 3] | | |
| Arrival rate $r^{arr}$ | [0.5, 0.5] | | |
| Execution penalty $p^{exec}$ | [4, 1] | | |
| Off-loading penalty $p^{off}$ | [2, 2] | | |
| Number of adversaries K | 1 | | |
| Defense Algorithm | δ = 0.1 | δ = 0.4 | δ = 0.8 |
| Q-learning | 6 | 3.5 | 0.8 |
| minimax-Q | 6.2 | 4.3 | 2.0 |
| RoM-Q | 6.8 | 5.7 | 2.8 |

Table 4. Test performance under rom-q-based attacks

| Defense Algorithm | δ = 0.1 | δ = 0.4 | δ = 0.8 |
|---|---|---|---|
| Q-learning | 6 | 3.6 | 1.6 |
| minimax-Q | 6 | 3.8 | 2.4 |
| RoM-Q | 6.7 | 5.3 | 3.0 |

By $r^{arr}$, the number of tasks on a node will increase by 1, and the node will execute actions ai, ai, or not doing anything. Then, the condition will be updated and check whether the task number exceeds $c_i$, and determine whether the $p^{repr}$ will be given. In the end, the sum of rewards minus the sum of punishments will be the result. The parameters if the experiment is shown in this table:

According to the table, RoM-Q learning is not only performing the best in all the defending conditions, but also having the best attack effect, since the RoM-Q attacking environment makes all three algorithms to have lower performing scores, while Q-learning performs poorly in attack-existing environment, and though minimax-Q performs well in both conditions, it still falls short comparing to RoM-Q.

The advantage of this paper is that it posts many pseudocodes and graphs to explain the algorithms, and it posts the results of the experiment by graphs, which seems clear to readers. The idea that “prepare for the worst” in RoM-Q learning provides the algorithm the ability to perform well in multi-agents attack environment. However, there still exists many weaknesses, such as that the authors didn’t explain the resulting graphs, so the readers have to go back to the previous part to

DOI: 10.54254/2755-2721/2025.PO25141

check the variables or data. Also, the variety of factors of the experiments could be improved by including more ways of policies other than just Q-learning, minimax-Q, and RoM-

Q. RoM-Q's ability in attacks and defenses can be more explored by testing in different multi-agent environments. Last but not least, the data in the graphs do not have exact values shown, so the readers can only observe the trend of the algorithms' performance, and estimate the result values. This forces readers to take more steps in understanding and comparing the data, which is an important disadvantage of the paper.

In a conclusion, RoM-Q is an essential algorithm in facing attacks, according to the experiment, and its ability to pre- pare for the worst situation allows the algorithm to perform influentially under multi-agent system.

## 5. Playing FPS games with deep reinforcement learning

[3] modified a model based of DRQN for training AI for play First-Person-Shooting (FPS) game (Doom), which performance well in both full death match and limited death match.

The model is principally based on deep recurrent Q- networks (DRQN). However, DRQN has certain disadvantages. It performs well in simple conditions (e.g. turning or attacking), but not in more comprehensive conditions. It may continue to shoot and wait for enemies to move to their fire line. One reason for this is that it may not be able to detect enemies. They used a ViZDoom engine. This engine gives the access to the internal variable generate by the game. Based on this, a Boolean value which recording weather there is an entity or not is used. Only the frame with entities will be used for further model. The game then goes through a convolutional neural network, and the output is then split into two different layers as some will give to LSTM for normal DRQN model, others marked as game features, an extra hidden layer. This layer is training with Q-learning object. This game feature improves nearly twice the kill-die ratio than that not has.

The whole game task is divided into two parts: the navigation and the action phase. This gives out several advantages. First, training two network together is faster than training one. Second, the navigation phase only requires three actions, so it reduce the state-action pairs required and faster. The two phases are training by different method. DRQN for action phase when enemy detected and DQN for simple navigation, when no ammo or enemies. The switch between different method also increases the performance [4-6].

For the training part, the first part is reward. The score getting in the game require the agent to do a list of action, so it is difficult for agent to learn which action refers to it. To solve this problem, the reward shaping, that include small intermediate reward in training to speed up the process is used. Three rewards are added to action phase, which are:

1) Positive reward on picking up objects
2) Negative reward on injured
3) Negative reward on losing ammo

Also, some rewards are added to navigation:

1) Positive reward on picking up item
2) Negative reward on wrong route
3) Positive reward on shorter distance

Another process called frame skipping is used. The frame skip means that the agent will only receive the K+1 frame, As K is the frame it skips. The process can increase the training efficiency but may harm the performance. They choose to skip 4 frames for training to reach a balance of accelerating the training and the hurt on performance.

DOI: 10.54254/2755-2721/2025.PO25141

Sequential updates are also be used. N observation is randomly sampled but only the one that have enough history (hidden state) will be updated. They selected that only observation with four histories will be used as a total of eight history. This appropriate update leads to a higher kill-die rate during the training. If there are more updates, it will cause a high correlation in sampled frames. Or there is less updates, it will be difficult for network to train.

For experiences, starting with hyper-parameters, the training using RMS-Prop-algorithm and mini-batches of size 32. The screen uses a 16/9 resolution of 106*60, this allows the agent has a 108-degree vision. Grayscale images lead to a low performance, so colors image is used.

The whole experiment is done on ViZDoom platform, as fight against Doom bots. The score is regarded as the killing numbers. Two competition is been tested, one is the limited death match, which is on a specific map with specific weapon. Another is the full death match, which means experiment on random map with different weapons.

The evaluation metrics is the kill-die ratio. The number of items picked up is also included. These let the agent prevent dying and encourage it to explore.

For the result, the data was recorded during a 15 minutes play. To begin with the effects of navigation in limited death match, the object picked is three time as more as the one with no navigation, at 46 and 16 respectively. However, it has limited effect on K/D ratio, as the map is relatively small and more enemies that finding enemies is not crucial. However, is has a significant effect on full death match, where the K/D rate enhanced from 3.12 to 6.94 and also picking up more objects. The agent behavior is even better than humans. As the K/D ratio of agent is nearly five time as more as human and perform well on fighting with each other. It kills human 8 times during a 5-time fight compared with human only killed it 5 times. The suicides number is also lower for agent than human, which means it has less errors on playing.

For game features, it rise the K/D ratio to more than 4, compared with the one with no game feature in model only reached 2 at the same training time. Another advantage is that it gives feedback on the quality of each frame. So, the LSTM model will only receive the one with high enemy detection accuracy. The enemy detection accuracy takes a week to train but reasonable as it highly related to the final performance of the model. The accuracy with and without dropout is also different at 90% and 70%, which also improve the performance of the model.

For conclusion, this model has some improvements and these improvement of this model leads to a significant improvement as it has a high K/D ratio and also better than real human players.

## 6. Conclusion

In conclusion, the review of these three topics highlights the significant advancements in the application of Q-learning algorithms for AI in computer games. FAQ Learning provides a novel approach to address the challenges posed by multi- agent learning environments, enhancing the robustness and efficiency of AI systems. RoM-Q Learning demonstrates how its ability to defend against multiple attacks environments is better than the other algorithms. Lastly, DRQN shows impressive performance in FPS games, outperforming human players in terms of kill-die ratios.

Although there are some inaccuracies in the performances of AI in games, we still believe that there are some improvements in AI performances. Collectively, these studies underscore the importance of developing advanced learning algorithms for AI in games. As we continue to explore the boundary of artificial intelligence, the insights gained from these studies will undoubtedly contribute to the development of more sophisticated and robust AI frameworks for gaming applications. The future of AI in games looks promising, with the potential for even greater advancements in performance and adaptability.

DOI: 10.54254/2755-2721/2025.PO25141

## Acknowledgment

We held sincere gratitude to Professor William Nace for his teaching of knowledge in courses and the guidance on this review paper. Professor Nace's teaching of algorithms in video games takes us deeper into the world of coding, and his support and suggestion greatly helps the process of writing this paper. Without Professor Nace's assistance, we wouldn't have done this paper smoothly. Professor Nace's guidance will not only help us in this paper, but also influential to all kinds of works in the rest of our life.

## References

[1] Michael Kaisers and Karl Tuyls. Frequency adjusted multiagent Q- learning. In Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems, pages 309–316, 2010.
[2] Eleni Nisioti, Daan Bloembergen, and Michael Kaisers. Robust multi- agent Q-learning in cooperative games with adversaries. Autonomous Agents and Multi-Agent Systems, 35(1): 1–23, 2021.
[3] Guillaume Lample and Devendra Singh Chaplot. Playing FPS games with deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 31, pages 2140–2146, 2017.
[4] Chris Gaskett, David Wettergreen, and Alexander Zelinsky. Q-learning in continuous state and action spaces. In Advanced Topics in Artificial Intelligence, volume 1747 of Lecture Notes in Computer Science, pages 417–428. Springer, 1999.
[5] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. Q-learning algorithms: A comprehensive classification and appli- cations. IEEE Access, 7: 133653–133667, 2019.
[6] Richard S. Sutton. Learning to predict by the methods of temporal differences. Machine Learning, 3: 9–44, 1988.