Comparative Analysis of Frozen lake Games for Selection of Optimal Action and Reward Using Q-learning, SARSA and Dynamic Programming Techniques

Harsh Kumar
Dept. of CSE
IIIT-Naya Raipur
Chhattisgarh, India
Email:harsh19102@iiitnr.edu.in

Harsh Chandrakar
Dept. of CSE
IIIT-Naya Raipur
Chhattisgarh, India
Email:harsh19100@iiitnr.edu.in

Santosh Kumar
Dept. of CSE
IIIT-Naya Raipur
Chhattisgarh, India
Email:santosh@iiitnr.edu.in

Mithilesh Kumar Chaube
Dept. of Mathematical Sciences
IIIT-Naya Raipur
Chhattisgarh, India
Email:mithilesh@iiitnr.edu.in

Anurag Singh
Dept. of ECE
IIIT-Naya Raipur
Chhattisgarh, India
Email:anurag@iiitnr.edu.in

***Abstract*—Reinforcement learning is gaining more proliferation in various applications such as traffic control, self-autonomous vehicle, robotics, and game playing. This article develops an optimization method deployed on SARSA, Dynamic programming, and Q-learning techniques. The proposed method allows learning the Q-learning agent to measure its state's expected rewards and utility to make optimal actions in an unknown dynamic environment. This method provided a comparative study of reinforcement learning and exploration in algorithms and efficacy of the system in popular games such as frozen-lake. Q-learning algorithm is used to train an agent to play the actual game, such as frozen-lake. We compared results obtained by different algorithms and find the most optimal one for the environment. Based on the complicated created dynamic environment agent performed well in the SARSA algorithm.**

***Index Terms*—Dynamic Programming, Q-learning, Reinforcement learning, SARSA**

I. Introduction

In Recent days reinforcement learning played a vital role in the development of different AI techniques. Different conventional approaches for developing games also suffered for different kinds of problems such as to solve these problems different researchers' proposed different interactive environments to solve this problems [1].

In RL, the agent learns to control itself from obstacles placed on the surface without getting affected. Recent advancement in-game automation has made it possible to solve such a problem in a very effective manner. However, it presented some challenges there are some drawbacks of every algorithm similar to SARSA algorithm works on a low-risk low reward basis [2]. On the other hand, Q-learning used the greedy approach for selecting the optimal path for the computation of rewards. However, it may fail to provide better results output. Dynamic programming techniques explore and every path and store results in some variables, which will cause higher complexity [3].

On the other hand, Q-learning used the greedy approach for selecting the optimal path for the

computation of rewards. However, it may fail to provide better results output. Dynamic programming techniques explore and every path and store results in some variables, which will cause higher complexity [3]. Our agent successfully learns to control the environment with different algorithms available and tries to overcome the existing problem of game automation.

II. Motivation and Contribution

Here we compare three different algorithms to solve this problem based on the result obtained from different approaches. We are intrinsically motivated to solve the reinforcement learning problem as it is an emerging field in artificial intelligence. We can apply reinforcement learning techniques to different branches of artificial intelligence, such as deep learning, robotics, game playing, and natural language processing techniques [4].

The major contributions are illustrated as follows:

1) This paper provides a comparative study of different algorithms for analysis of frozen lake environments, the environments provided by the open-AI gym. It is an open souse environment for training the operator and analysis of results. This paper makes no prior assumption related to the formation of the operator and is comparable with any computation library, such as Tensor Flow or Theano Frozen Lake-v0.
2) At first, we applied a dynamic programming technique based on the result obtained. We compared it with the Q-learning technique to calculate the maximum rewards based on selected actions. We further compared the results with the SARSA algorithm. Our main aim in this paper is to solve the frozen-lake problem with a lesser number of episodes and at the same time with a higher number of accuracy. Frozen-lake is a grid-based game consisting grid of size 4×4. An agent has to choose from four possible ways up, down, left, and right.
3) Our operator commands the motion of features in a framework. Some block of the framework are movable, and other leads some absolute and ultimately falling inside water. Moreover, the movement path of our operator is not pre-defined. It depends on the path which our agent has chosen while iterating. If the operator manages to reach the final destination reward will be awarded to the operator for finding the walkable path.

III. Preliminary

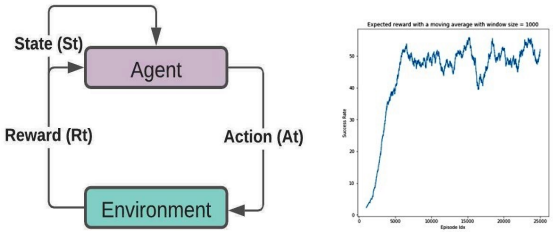
The given figure agent takes action in some environment and produces some reward Rt+1 and some other state St. This process continues till the episode ends. In this section, reinforcement learning techniques and other definitions and symbols are given in the following subsection.

A. Reinforcement learning

It is a branch of machine learning in which agents try to learn by performing different actions Shown in Fig. 1(a). Every action contains some reward if the agent gets a positive reward. It contains its action if it gets a negative reward, then it backtracks from the particular action and explores different actions. Here our operator aims to maximize profit as experience increases over time. Mainly there are five main elements of the reinforcement learning model [1]- [6].

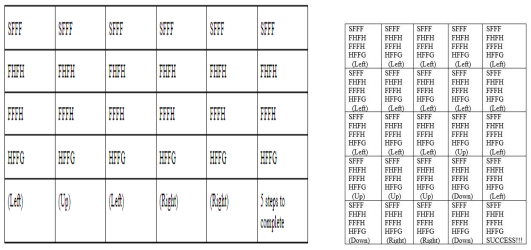
1) Agent: The active learner interacts with environment
2) Reward: The reward or reward function calculates the rewards based on given actions.
3) State: It shows the movement of the agent in the defined environment.
4) Action: Actions are performed by an agent to perform the tasks.
5) Environment: Agents interact with the environment.

The agent performs control on several movements in the given state of a grid. Various grid blocks are defined in the form of movable matrices. Moreover, the motion command of the agent is not known. The agent is only able to learn selected conditions. To maximize reward, the agent finds the suitable path to a goal tile [5].



(a) (b)

Fig. 1. Shows (a) working of reinforcement learning system., (b)Mean reward over a time interval.



(a) (b)

Fig. 2. Shows (a) walking pattern of the agent(b) π Episode 0 agent movement.

The solution must be in the form of a learning algorithm (Jupiter notebook) to learn to reach a goal after a series of training exercises. The surface is described using the below grid: (1) SFFF, (2) FHFH, (3) FFFH, and (4) HFFG. The symbol s indicated the beginning point of the agent, the frozen surface is defined by (F), and the hole is defined by (H), where the agent may/may not fall inside the hole, and g stands for a goal where some landmark is located at the target point. The event ends when the agent reaches the destination or spills inside a hole. Agent receives a reward of 1 if it reaches the destination and zeroes otherwise. The agent can take the path as 1. LEFT, 2. DOWN, 3.RIGHT, and 4. UP [6].

B. Depicting one attempt

We tested our proposed model to check action performed by the agent based on the provided walking pattern. We used different algorithms such as dynamic programming to find the optimal path for an agent for depicting one attempt. The agent takes the optimal path to perform actions for the reward obtained. In the practical scenario, the agent can not reach the goal at specific steps taken [7]. Figure 2 (a) shows the walking pattern of the agent and the steps it takes after every episode. At the final step, it reaches to destination, and the total steps count is five.

C. Training of the Agent

Some important points that we should know before training an agent:-

1) s: current state in which agent is present.
2) a: current action which agent picked for some policy (π).
3) s': s' indicates the movement of the agent to the next state.
4) a': the action performed by the agent.
5) r: reward of action taken.
6) α:learning rate of agent.
7) γ :discount factor for future reward.

D. Random walk

In this paper, we used the ϵ-greedy-based probability method to calculate rewards. The agent ends up at the 'GoalGoal if he randomly selects the corrected path and optimal actions in the right directions. We have computed probability and analytically. If the ice were not slippery, the actions were deterministic success percentage is 1.37%. The average number of steps taken to reach the goal =is 7.65.

E. Training a Q-Table for Slippery Conditions

In this section, to validate our model, we considered two cases: (1) the agent can complete the path successfully, and (2) the agent can slip on the surface. Q (Quality) the table is a num-state x num-action dimension table obtained by training on the map. Each state stores values proportional to the reward that can be obtained by taking each of the num-actions actions. In other words, if we were to have a fully trained Q-table, then the optimal action to take when we are in the state would be argmax (q-table [I,:]). [8] If the action was deterministic, then any action leading to a hole should store 0, and an action leading to the goal would store a large value [6]. Every episode, we update this table by observing the reward that is obtained by taking that step as below (shown in Eq: 1):

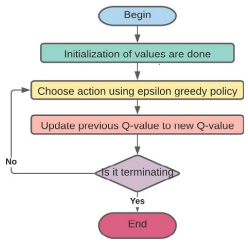


Fig. 3. Shows working of Q-learning system.

$$Q(s,a) = Q(s,a) + a*(R(s,a) + \gamma max'_a Q(s',a') = Q(s,a)) \quad (1)$$

Figure 2 shows the movement of agents and the updated result obtained based on calculated mean rewards over a time interval shown in Fig. **??**.

1) Q(s, a) = indicates the value given in a Q-table. wheres and a are the state and action.
2) R (s, a) = reward obtained by acting a form state s = Discount Factor used to balance immediate and future reward
3) s' = next state movement by the agent.
4) a' = Maximizing over all actions from the destination state s.' Observations
5) During movement from the previous state to the next state, some states have indicated value 0 such as state(S) = [5,7,11,12 and 15]. We land up in this state. While states (S)=[5, 7, 11, and 12] states correspond to 'Hole' where the game ends with 0 rewards, state 15 corresponds to 'Goal,' which ends with one reward. Let us plot the mean reward over time and see how our agent has learned over the training period. [9]
   Figure 1(b) shows the mean reward of the agent over a given time interval for every episode count. We can observe that the mean reward is lower at starting episode, but as the agent moves to the next episode, the mean reward increases gradually over time. The total episode count is 25000, and the success rate is between 0 to 50 for an episode. [10]

Figure 2 illustrates the movement of the agent for particular episode 0 and the actions it takes for moving to the next step. There are two possible ways for the movement of agents on the grid. We consider both ways as case 1. and case 2.

1) **Case 1:** the agent moves on the surface where the obstacles are present. The average number of steps taken to reach the goal is equal to 44.53. The agent's success rate using this Q-Table is 74.6%. Notice that the average number of steps taken to reach the goal is very high. The agent takes many conservative steps (for example, moving UP even though there is a boundary wall) to prevent slipping into the hole. Overall success percentage = 74.57% [11].
2) **Case 2:** The agent's movement on the surface where the obstacle is not present an interesting case to consider would be to assume that the ice was not slippery. In such a case, the agent would directly want to head to the goal. What would the success rate be if we were to adopt that algorithm wherein the agent ignores the fact that the ice is slippery and blindly heads towards the goal (of course, on the ice and avoiding the hole)? We obtained a success percentage is 4.11%. The average number of steps taken to reach the goal = 5.03. We see that the success percentage using this q-table to take the optimal path if the ice were not slippery has an abysmal success rate. Also, we end up falling into the hole (or reaching the goal with a 4%) within an average of 5 steps. [12]

IV. Proposed Method

This section proposed an optimization method based on the Q-learning technique to maximize agent reward based on the accuracy and a minimum number of steps the agent takes to reach the final target in a particular environment. [2] [3] The proposed method is used to solve the maximization of frozen-lake gaming problems include deep Q-learning, dynamic programming, and SARSA algorithm. These algorithms are used to calculate the rewards of the given problem. Our agents have used a dynamic programming algorithm that stores the result obtained in a two-dimensional array to avoid

Algorithm 1: Iterative policy evaluation

1) Initialization: input policy (π), threshold (θ) $\geq$ 0). It measures the accuracy of reward estimation.
2) V(s) is defined for all $s \in$ S. We assumed that arbitrarily expect that V(terminal) = 0.
3) Objective is to estimate the policy evaluation

while *While condition* **do**

Loop: $\Delta \leftarrow 0$ Loop for each s $\in$ S: V $\leftarrow$ V(s)

$$v(s) = \sum \pi(a|s) + A \tag{2}$$

where A

$$A = \sum s', rp(s', r|s, a)[r + \gamma V(s')] \tag{3}$$

$$\Delta l \leftarrow max(\Delta, |v - V(s)|) \tag{4}$$

Unit $\Delta \leq (\theta)$ Iterative policy evaluation, for estimating V = v_π

end

Result: V=v_π

recalculation in overlapping sub-problem. A brief description of these algorithms are given below:

1) **SARSA:** Sarsa algorithm is the modified version of the Q-learning algorithm. It is an on-policy algorithm. It learns the Markov decision process with policy based on the agent's interaction with the environment and computes the value of action taken by the agent. It updates the policy based on Q-learning's previous action: it takes a greedy approach to solve the problem. It first chooses the path with maximum reward, but sometimes it may lead to a nonoptimal answer. It chooses new actions based on the existing highest reward. [4]
2) **Q-learning:** is an off-policy method to solve complex problems. It is an emerging technology in reinforcement learning. Q-value is updated based on the off-policy method. It follows a greedy path at the time of model training. It uses maximum operation while estimating the algorithm's future reward greedy path allows the operator to move randomly. In this process, the agent explores new blocks and experience new possibilities. The advantage of Q-learning over other on-policy algorithms is that Q-learning does not get trapped inside local minima. Store the Q- table and initialize the frozen lake environment. Then different variables were introduced epsilon (ϵ) for the greedy approach. The game is for discount factor, max-episode is the maximum number of times a game will run, max step is the maximum number of steps agent is taking to reach the goal, (lr_rate) learning rate of the agent. Initialize Q- table with zero values agents recursively chooses the action which gives maximum reward using a greedy approach for different states (shown in Algorithm **??**).
3) **Dynamic Programming:** In dynamic programming, we find the optimal policy for the agent. Mainly dynamic programming problem is divided into two main parts:
   a) Breaking down the given problem into a smaller set of problems.
   b) Storing the problem in an array for future use purposes.
   c) This method evaluates our policy by computing the state value function using the updated bellman expectation equation.

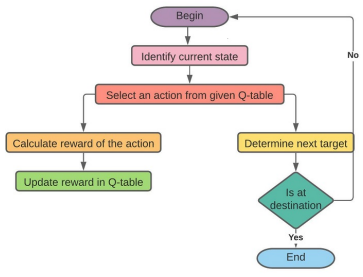


Fig. 4. Shows working of Q-learning system.

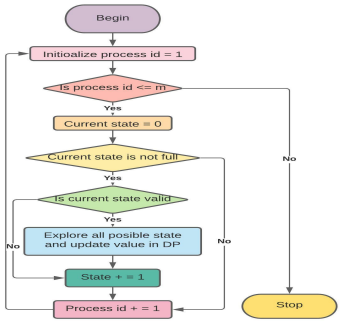


Fig. 5. Working of Dynamic programming system.

$$V_{k+1}(S) = E_{\pi}[R_{t+1} + B \qquad (5)$$

d) Where B

$$B = \gamma v_k \times S_{t+1}|S_{t=s} \qquad (6)$$

The shortcoming of dynamic programming over the Q-learning technique is that dynamic programming is a tabular solution method used to compute optimal policy in a given Markov decision process because of their assumption and computational expense of perfect models. [5]

A. *Problem formulation*

This paper compares three different techniques to solve a grid-based reinforcement learning game called frozen-lake. Here learning agent is trained to interact with the environment provided by an open AI-gym. We created the environment for depicting Frozen Lake. It is a game consisting of a grid of dimension 4×4, which contains four possible states: A starting point(s), ending point(g), obstacle(h), and frozen surface(f). Our agent tries to reach the goal it faces some obstacles. If the agent manages to bypass all the obstacles, it will get +1 as a reward. If it falls inside the hole, it has to start from the starting position and is rewarded the value 0 [13]- [14].

The grid is a 4×4 square matrices. Our agent is initially at the position (0, 0), and it has to travel to (n-1, n-1) to reach the goal bypassing the obstacles placed randomly in the grid. The starting position is represented by (S), the goal is represented by (G), the frozen surface/walkable path is represented by (F), and the obstacle/hole is represented by (H).

Algorithm 2: Algorithm for Computation of Reward

1) Initialization: current state s,
2) select action from Choose an action a from the set consisting of available welding seams along with all possible directions.
3) these are not welded yet which provides minimum of the max. structural deformation
4) a = arg min a' A,[Q(s',a')] with greedy policy
5) Q(s', a') = minimum of the maximum structural deformation= 0.2].
6) Execute action a;
7) Update Q-value with Q(s, a)- min a,[Q(s', a')]
8) remove a and - a from A'.
9) Update state translation model p(s'—s,a);
10) Store S = s and A = A, a
11) End
12) End S, A: states and actions consisting of the stochastic shortest path.

Result: Return stochastic shortest path

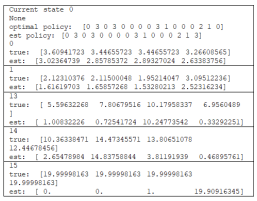
V. Experimental Result and Analysis

In this section, we analyze the performance of the proposed method with different parameters.

A. *Result*

In this section, the experimental result is provided based on different settings, such as environmental parameters. Fig. 6 is showing the performance of the agent for different episodes on applying the SARSA algorithm. It is also showing the actual outcome vs. the estimated outcome of the agent for the SARSA algorithm.

Figure 6(a) is showing the performance of the agent for different episodes on applying the SARSA algorithm. It is also showing actual outcome vs. the estimated outcome of the agent for the SARSA algorithm. Figure 6(b) is showing the performance of the agent for different episodes on applying dynamic programming algorithms. It also



(a)

Episode: 0	Reward: 0.0	Steps Taken: 4
Episode: 120	Reward: 1.0	Steps Taken: 12
Episode: 310	Reward: 1.0	Steps Taken: 10
Episode: 440	Reward: 0.0	Steps Taken: 4
Episode: 570	Reward: 1.0	Steps Taken: 8
Episode: 890	Reward: 1.0	Steps Taken: 6

(b)

Fig. 6. Shows (a) Result obtained for SARSA algorithm at different states, (b) Dynamic programming algorithm at different states.

shows episode count, reward, and steps taken by the agent for the Dynamic programming algorithm. Fig. 7 is showing the performance of the agent

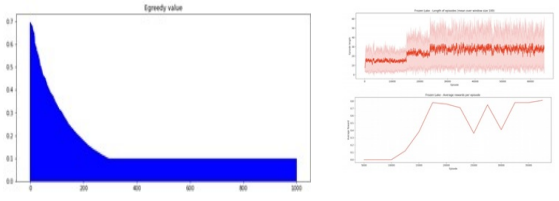
```
Current state 0
None
optimal policy:  [0 3 0 3 0 0 0 0 3 1 0 0 0 2 1 0]
est policy: [0 3 0 3 0 0 0 2 0 3 1 0 0 0 2 1 2]
0
true:  [3.60941723 3.44655723 3.44655723 3.26608565]
est:  [3.85528511 3.753742   3.70045909 3.4882637 ]

14
true:  [10.36338471 14.47345571 13.80651078
12.44678456]
est:  [10.47033934 14.95846553 13.16000832 12.29886623]

15
true:  [19.99998163 19.99998163 19.99998163
19.99998163]
est:  [19.69536126 19.66807237 19.78324759 19.69043457]
```

Fig. 7. Result obtained for Q-learning algorithm at different states.

for different episodes on applying the Q-learning algorithm. It is also showing the true outcome vs. the estimated outcome of the agent for the Q-learning algorithm.

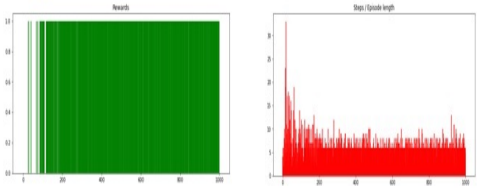


(a) (b)

Fig. 8. Shows (a) characteristics of ϵ-Greedy values for selection optimal actions, (b) length of the episode (top), shows average reward per episode (bottom).

B. Analysis

Based on overall observation, we analyzed three potential algorithms that the agent can take to reach its goal-the path he takes through the random walk with a success rate = 1.4%. An agent takes a total average number of steps is equal to eight. The Q-table is trained on slippery conditions for better calculation of performance of the model, and the performance is reported with a success rate of 75% with the required average number of steps is 45. On other setting conditions based on Q-table trained on non-slippery condition cases, the success rate has been reported as 4% with the required average number of steps is 5.

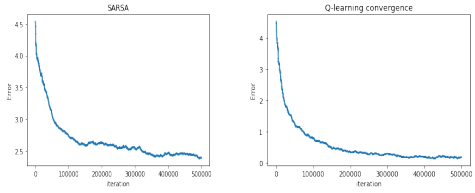


(a) (b)

Fig. 9. Shows (a)reward measurement for different actions, (b) steps and the episode length for agent movement.

Figure 9 (a) shows the reward of the agent from 0 to 1 for each episode. The measured reward for different actions is used. It is used for performance measurements taken by different actions.

Figure 9 (b) shows the taken steps count of agents per episode to reach the goal. The agent has taken the total step count based on the total number of episodes taken by the agent. Based on given Figure 9, the step count is required at starting episodes for calculating the optimal reward to reach a defined goal. Once the model is trained based on



(a) (b)

Fig. 10. Shows (a) SARSA algorithm error vs. iteration graph, (b) Q-learning error vs. iteration graph.

selected optimal actions, the learned agent takes the minimum number of steps to count based on the ϵ-greedy search method for optimal episodes.

Figure 8(a) depicts that steps of the agent in starting phase were significant due to high random action in starting and later on. As the agent is used to learn from dynamic experience, it starts to perform actions and more informed steps. Hence more minor error (noises) is reported. Figure 8(b)shows the total length of the episode (top). The bottom shows the average reward obtained per episode by the agent. [4]

Figure 10 (a) shows the measured error rate of the algorithm vs. the number of episode counts by the SARSA algorithm. It depicts error rate is decreases as the number of iterations/epochs increases. The error rate is high at 10^5 epoch. It is reported that the error rate is a minimum of 2.5 at several taken iteration 5×10^5. Figure 10 (b) shows the error rate of the different algorithms vs. the number of episode counts by the Q-learning algorithm.

VI. Conclusion and Future Direction

In this paper, Q-Learning technique-based frozen-lake game implementation is illustrated. We compared the performance of the model using different algorithms such as SARSA, and Dynamic programming, and Q-learning algorithm, which uses the off policy method. The people may apply this algorithm to train their prototype using unsupervised learning. As soon as the agent memorizes the state, our model meets at a point where random steps are not required. Further, we can track down our operator at each step and every iteration of events. By applying these examples, we learned different algorithms and its concept related to Reinforcement learning-every time we want to put reinforcement learning in any problem. We want to solve optimization for obtaining a maximum reward for the agent. We have to establish our environment, identify the possible action that our agent can take, determine different states and set a reward for every action that the agent will take.

References

[1] Politowski, C., Petrillo, F., Ullmann, G.C. and Guéhéneuc, Y.G., 2021. Game industry problems: An extensive analysis of the gray literature. Information and Software Technology, 134, p.106538.

[2] I. Oh, S. Rho, S. Moon, S. Son, H. Lee, and J. Chung, "Creating Pro-Level AI for a Real-Time Fighting Game Using Deep Reinforcement Learning," in IEEE Transactions on Games, DOI: 10.1109/TG.2021.3049539.

[3] Padakandla, S., 2021. A survey of reinforcement learning algorithms for dynamically varying environments. ACM Computing Surveys (CSUR), 54(6), pp.1-25.

[4] Stanford, F.S., 2003. Escaping the frozen lake: individual and social idealism manifest as forms of religion and religiosity. Texas A&M University.

[5] Lindgren, Prajna, Guido Grosse, Franz J. Meyer, and Katey Walter Anthony. "An object-based classification method to detect methane ebullition bubbles in early Winter lake ice." Remote Sensing 11, no. 7 (2019): 822.

[6] Sharma, Sapna, Michael F. Meyer, Joshua Culpepper, Xiao Yang, Stephanie Hampton, Stella A. Berger, Matthew R. Brousil et al. "Integrating perspectives to understand lake ice dynamics in a changing world." Journal of Geophysical Research: Biogeosciences 125, no. 8 (2020): e2020JG005799.

[7] Yang, Xiao, Tamlin M. Pavelsky, Liam P. Bendezu, and Shuai Zhang. "Simple Method to Extract Lake Ice Condition From Landsat Images." IEEE Transactions on Geoscience and Remote Sensing (2021).

[8] Shao, K., Tang, Z., Zhu, Y., Li, N., and Zhao, D., 2019. A survey of deep reinforcement learning in video games. arXiv preprint arXiv:1912.10944.

[9] Mozgovoy, M., Preuss, M. and Bidarra, R., 2021. Team sports for game AI benchmarking revisited. International Journal of Computer Games Technology, 2021.

[10] Tang, H., Wang, A., Xue, F., Yang, J. and Cao, Y., 2021. A novel hierarchical soft actor-critic algorithm for multi-logistics robots task allocation. IEEE Access, 9, pp.42568-42582.

[11] Torrado, R.R., Bontrager, P., Togelius, J., Liu, J. and Perez-Liebana, D., 2018, August. Deep reinforcement learning for general video game ai. In 2018 IEEE Conference on Computational Intelligence and Games (CIG) (pp. 1-8). IEEE.

[12] Pfau, J., Liapis, A., Volkmar, G., Yannakakis, G.N. and Malaka, R., 2020, August. Dungeons & replicants: automated Game balancing via deep player behavior modeling. In IEEE Conference on Games (CoG) (pp. 431-438).

[13] Pfau, J., Liapis, A., Volkmar, G., Yannakakis, G.N. and Malaka, R., 2020, August. Dungeons & replicants: automated Game balancing via deep player behavior modeling. In 2020 IEEE Conference on Games (CoG) (pp. 431-438). IEEE.

[14] D'Souza, S., 2021. Implementing Reinforcement Learning Algorithms in Retail Supply Chains with OpenAI Gym Toolkit. arXiv preprint arXiv:2104.14398.