https://doi.org/10.26682/sjuod.2023.26.2.45

Journal of University of Duhok., Vol. 26, No.2 (Pure and Engineering Sciences), Pp 489 - 501, 2023

(Special Issue)

REINFORCEMENT LEARNING MODEL FOR FINDING OPTIMAL PATH

ARAN **S**IRWAN **I**SMAIL, **Z**HIAR **A**HMED **M**OHAMMED, **K**OZHIR **M**USTAFA **H**USSAIN, and **H**IWA **O**MER **H**ASSAN
University of Human Development- Iraq

(Accepted for Publication: November 27, 2023)

ABSTRACT

Reinforcement learning (RL) has become a powerful method for addressing complex optimization challenges, such as determining optimal (shortest) paths across various fields. This study aims to design and examine RL algorithms to identify the shortest path in expansive and dynamic environments. Our research advances RL algorithms for optimal pathfinding, showcasing Q-Learning's scalability, path length, and cumulative rewards superiority, enriching optimization methodologies, and guiding future RL explorations. We utilized a blend of Q-learning and Double Q-learning methods to improve our model's performance in conjunction with the Reward Shaping technique and exploration-exploitation tactics like epsilon-greedy with decaying epsilon. We evaluated our approach on a 27x27 matrix representing the environment. In both 17x17 and 27x27 environments, Q-Learning consistently showed optimal paths with shorter distances, while having a slightly slower performance than Dijkstra's algorithm, with completion times of (0.00093 seconds versus 0.000023 seconds and 0.0097 seconds versus 0.000034 seconds); nevertheless, our assessment encompassed Q-Learning, Dijkstra, and Random-selection algorithms. Q-Learning consistently showcased excellence, while random selection yielded suboptimal routes due to its inherent randomness. Cumulative rewards highlighted Q-Learning's superiority (ranging from 2,197,570.15 to 205,823.20), reflecting its adept learning capacity. Dijkstra prioritized minimal path length (cumulative rewards ranging from 6 to 1), whereas Random-selection exhibited considerable variation (ranging from 132,449 to 11,901).

KEYWORD: Reinforcement Learning, Shortest Path Finding, Q-Learning, Double Q-Learning, Reward Shaping, Decaying Epsilon and Alpha.

1. INTRODUCTION

RL begins a mission for quick decision-making within complex scenarios. In modern computer science, RL emerges as a foundation [1]. It empowers computational agents with understanding, essential for addressing complex scenarios and gaining favorable outcomes within specified contexts [1]. Unlike conventional learning models, RL divides from data memorization. Instead, it reflects the iterative process of trial and error, like human learning [2]. Imagine agents incorporating knowledge through their actions, similar to our learning [2]. However, RL is more than a learning mechanism; it is a dynamic adaptation process reflecting human learning journeys [2]. Within this scope of RL, the Q-learning algorithm assumes center stage. It is comparable to producing two strategies: one for action selection and another for insights from outcomes. This balance between exploration and exploitation evolves over time [3].

In the realm of RL, two distinct categories emerge: one entails experimental learning (model-free Q-learning), while the other leverages existing knowledge to guide decision-making (model-based strategies) [4]. The captivation of Q-learning lies in its strength during uncertainties. Its ability has enlightened fields such as robotics, autonomous vehicles, and aerial drones [4], [5], [6], [7]. At the heart of RL resides the journey between agents and their surroundings, formed by rewards that signify action effectiveness [8]. Through cumulative interactions, agents refine their decision-making ability, improving their proficiency in goal achievement [8]. This journey ends as agents accumulate optimal actions [8].

Our study is about RL's potential to revolutionize decision-making across different landscapes and focus on RL's role in separating the complexities of pathfinding. We synergize Q-learning and Double Q-learning methods, improved by Reward Shaping, Decaying Alpha, and customized exploration strategies. This

experimental framework is tested within a 27x27 matrix against conventional methods. Our effort examines RL's pathfinding performance, charting the path for future research and real-world applications.

The paper is structured as follows: Background and Research Gap contextualize navigation complexities and innovation through RL. Literature Review explores RL's role in optimal pathfinding and its connections to diverse applications. Methodology describes the utilization of Q-learning, Dijkstra's algorithm, and the Random-selection algorithm. Evaluation Approaches analyze performance metrics for Q-learning, Dijkstra, and Random-selection algorithms. Metrics deepens the understanding of algorithmic strengths and limitations. Conclusion summarizes findings and implications. Future Work highlights unexplored areas of research and development.

BACKGROUND AND RESEARCH GAP

Navigating various scenarios, encompassing fields such as robotics, control systems, and transportation, has long posed an ongoing problem [1][4]. While conventional methods like graph search and dynamic programming offer support, their essential limitations come to the forefront when opposing complex and unpredictable environments [2]. As the complexities of modern pathfinding continue escalating, following creative methodologies becomes increasingly essential. In this context, the emergence of RL has garnered significant attention [1][2]. Agents equipped with the capacity to learn through interactions with their environment, displaying remarkable adaptability and ability to navigate changes [3], [5]. The characteristic adaptability of RL shows capability across a broad range of applications, spanning from autonomous parking [7] to the details of path planning in robotic navigation [9], [12].

The tempting promise of RL algorithms optimizing pathfinding strategies allows transformative implications across various domains, encompassing intelligent transportation systems and autonomous robotics. Nonetheless, the realm of achieving perfect pathfinding through RL is marked by persistent challenges, navigating them requires careful selection of algorithms from a range of available options [13]. Practical reward functions, essential for effective learning, become an imperative addition [9], [11]. Furthermore, the quality of environment simulations establishes the cornerstone for the success of RL actions [1], [3]. The ongoing scope of research activities is dynamically addressing these challenges, crafting reward functions that align with real-world objectives [7], and elevating the scalability of RL algorithms to tackle expansive issues [12]. An essential side of RL algorithms, enhancing sample efficiency, remains a dedicated research pursuit [12], thereby ensuring the similarity of real-world RL implementations with their inherent potential [7].

Subsequent research trajectories are composed to cover the development of robust RL algorithms proficient at navigating complex pathfinding scenarios [12]. Precisely calibrating reward functions tailored to specific applications further enhance the efficiency and effectiveness of RL [1], [9]. The journey towards elevated sample efficiency and advancements in environment modeling is anticipated to yield solutions characterized by their reliability and adaptability [2], [11]. The hidden promise of RL in addressing the continuing challenge of determining optimal paths within diverse environments is undeniably important. Conquering the challenges posed by algorithm selection, reward function design, and environment modeling assumes pivotal significance in unlocking the complete potential of RL for practical pathfinding applications [1], [9], [11].

2. LITERATURE REVIEW

RL has been widely used to find the optimal (shortest) path in various applications, including parking. Several studies have proposed RL-based approaches for different problems in parking, such as autonomous parking and parking lot management. These studies have demonstrated the potential of RL in improving the performance of parking systems and solving real-world problems. For example, a study proposed an RL-based algorithm for autonomous parking using Q-learning. This algorithm can autonomously park a vehicle in a parking lot while avoiding collisions with other parked cars [1]. Another study used RL to optimize parking in an intelligent city parking system by considering various factors such as the time of day, the day of the week, and the occupancy of the parking lot [7]. In another study approach, a RL-based end-to-end parking algorithm is proposed to achieve automatic parking; the vehicle can continuously learn and accumulate experience from numerous parking attempts and then learn the optimal steering wheel angle command at different parking slots [9].

Furthermore, studies have proposed RL-based approaches for parking guidance and control of electric vehicles and parking lot management. Recent studies have also applied RL to solving other path-finding problems, such as autonomous vehicle parking [9], unmanned vehicle navigation [10], electric autonomous vehicle path guidance [7], path planning and dynamic obstacle avoidance [11], and robotic path planning [12]. In the context of developing a RL model for optimal pathfinding, parallels can be drawn from challenges faced in stock market decision-making. Just as timing is crucial for investors, finding the shortest path is pivotal in navigation optimization. Similar strategies of informed decisions and mitigating impulsiveness in stock markets mirror the need for effective exploration and exploitation tactics in pathfinding algorithms [13].

In another study they used RL to train an autonomous driving agent in a simulated environment. The proposed approach performed better than the traditional path-planning methods [1]. Another study uses RL-based approaches for path planning in a 3D environment. The author compares a deterministic tree-based approach, a Q-learning-based approach, and an approximate policy gradient-based approach. The results show that the deterministic tree-based approach provides long-lasting results but has a higher computational time than the other two. The author concludes by comparing approaches to computational time and light search and then highlights the potential of RL for path planning, particularly in 3D environments where obstacles must be avoided for successful motion-path planning using Q-learning and other approaches [12]. Model-based and Model-free learning approaches are used to solve different problems. Model-based learning is typically preferred when the underlying system dynamics are simple, but the estimation of the model parameters is complex. In contrast, model-free learning is selected when the system dynamics are challenging, but estimating the model parameters is relatively straightforward. These approaches optimize performance criteria such as accuracy, computational efficiency, and stability.

In a study, authors found a safe and collision-free path for an agent from the start to the end of a designated area. It reflects the intelligence level of a mobile robot, and efficient algorithms can reduce loss. There are two types of path planning: static and dynamic, and two subtypes: local and global. Global path planning requires mastery of all map information and is often used for solving problems faced by regional path planning. Q-learning is a static path-planning algorithm with strong learning ability and adaptability to complex unknown environments. Q-learning-based path planning has become a hot topic in research due to the increasing demand for intelligent mobile robots [14]. The authors also introduce a priority weight into Q-learning, a RL algorithm, to improve its dynamic obstacle avoidance path planning. The improved algorithm is shown to have faster convergence speed and higher accuracy than previous algorithms. The importance of intelligent path planning algorithms in robot path planning and the challenges of dynamic obstacle avoidance are discussed.

The Q-learning algorithm is based on the time difference method. It has been widely used in path planning due to its ability to learn appropriate behavior without complete knowledge of the environment. The original Q-learning algorithm learns the optimal action-value function through a series of trajectories and updates the Q-value estimates through Monte Carlo approximation [14]. Another study developed an autonomous robot that uses Q-learning for navigation in a completely unknown environment; this will calculate the shortest path from the current state to the goal state by analyzing the environment through captured images. Further, the captured images will be processed through image processing and machine learning techniques, and the proposed method also takes care of obstacles present in that environment initially [15].

However, RL must address several challenges to find optimal (shortest) paths. These challenges include designing practical reward functions, scaling up algorithms to handle large-scale problems, and improving the sample efficiency of RL algorithms. Furthermore, the generalization of the RL algorithms to real-world problems is still an open research area. Overall, RL has significant potential to improve the performance of path-finding systems and solve real-world problems. Further research is necessary to address the existing challenges and extend RL's applications to more complex and dynamic environments.

3. METHOLODY

3.1 Finding optimal (shortest) Path

The problem involves applying RL techniques to determine the optimal (shortest) path for a

vehicle to navigate toward a designated parking spot. It entails devising a plan that leverages RL algorithms to guide the vehicle efficiently through a parking lot. Path planning is a computation problem that is an essential skill in robotics. The primary purpose of path planning is to find the best route between the start and the endpoint. For most robots, optimal path planning is the shortest path between two locations [15]. The critical aspects of this problem involve defining appropriate actions and rewards for the agent and training the agent to identify the optimal path amidst varying environmental conditions. These conditions may include blocked parking spots or changing traffic dynamics.
Fig 1

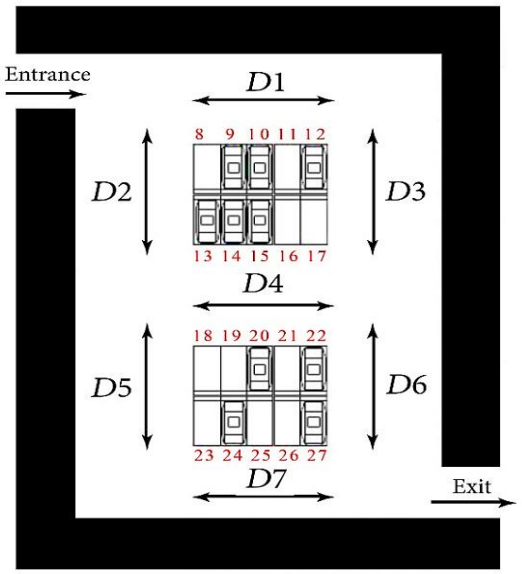


Fig 1

To tackle this problem effectively, enabling multiple agents to find the optimal path concurrently is crucial. This requirement highlights the significance of employing the Q-learning algorithm, which has gained popularity in RL. Q-learning is particularly suitable for solving problems necessitating determining an optimal policy for selecting movements. It offers various advantages, including handling challenges associated with extensive state spaces and delayed rewards. Using Q-learning to find the optimal path in a parking lot is particularly advantageous due to the inherent complexities involved. The number of potential states and actions in parking lots can be substantial, thus necessitating.

3.2 Visual Maps for Shortest Path Finding in Parking Lots

Our research project tackled challenges in designing the simulated environment and creating an appropriate reward matrix for our RL-based algorithm. The reward matrix guides the agent's actions and signals encouragement or discouragement. The authors pre-trained the Q-learning algorithm for path planning in a previous study. However, they faced difficulties in developing a reward matrix that accurately reflected the expected behavior of the agent in a complex parking lot layout [16]. To overcome this challenge, we meticulously devised a reward matrix with three distinct values: 1 for successful parking, -1 for unsuccessful parking, and 0 for all other actions and states within the simulation. We thoughtfully planned the construction of the reward matrix to incentivize successful parking attempts while penalizing unsuccessful ones, establishing a robust RL framework for our parking lot simulation.

The consideration of the reward matrix values played a paramount role in the success of our project. To comprehensively evaluate our algorithm, we devised seven paths that replicated real-world scenarios and typical traffic patterns in parking lots. The final map design closely resembled real-world parking lots, ensuring accuracy and realism in the simulation. Figure 1

showcases the custom-designed map used for our research purposes. Through planning and careful assessment, we successfully constructed a reward matrix that effectively facilitated the agent's learning process in the intricate parking lot environment. The integration of realistic scenarios and the design of the reward matrix contribute to the overall performance enhancement of our RL-based algorithm.

The infrastructure in the studied environment of Fig. 1. comprising of 27 nodes, can be categorized into roads and parking spots. The first seven nodes (1-7) represent the road network, facilitating efficient movement and transportation. The remaining nodes (8-27) are designated parking spots, ensuring convenient and accessible vehicle storage. This division optimizes traffic flow, enhances safety, and streamlines parking resource allocation. The Fig. visually represents the environment, accurately depicting labeled parking spots. The presence of labeled spots guides algorithm development and evaluation, allowing finetuning through iterative testing.

Integrating the elements mentioned above, including creating a reward matrix and incorporating realistic parking lot scenarios, has resulted in a comprehensive research project exploring the effectiveness of RL in parking lot navigation. Our findings and methodology offer valuable insights into the potential applications of artificial intelligence and machine learning techniques in addressing complex real-world problems.

3.3 Learning Algorithm

RL enthusiasts widely adopt and appreciate the effectiveness of this algorithmic strategy in addressing diverse problems across numerous fields [4, 11]. At the core of this approach lies the Q-values, which quantitatively measure the anticipated cumulative reward linked to executing a particular action within a specific circumstance and following the optimal path of action [17]. Q-values are pivotal in guiding the decision-making process, as they encapsulate a given state's quality or "goodness" regarding its potential value. When applied to navigating a parking lot, Q-learning provides a unique and effective technique for training an agent. The model of Q-learning uses a

$$Q(s, a) = Q(s, a) + \alpha (r + \gamma \max (Q(s', a')) - Q(s, a))$$

Where (s) is the current state, (a) is the current action, (r) is the reward for taking actions, (s') is the next state, (a') are the actions from the next state, (γ) is the discount factor to specify the priority of the coming rewards, and (α) is the learning rate which specifies how much the agent bi-dimensional reward matrix (R) as a reference representing the possible connections through trial and error. The rows are the states (s), and the columns are the actions (a) [4]. The agent actively explores the space of states, actions, and rewards to navigate the parking lot and reach the designated parking spot in the shortest possible time.

A Q-learning algorithm is a method of learning in interaction with the environment. Q-learning aims to learn strategies that tell agents what actions to take under what circumstances. It can deal with the problems of random transformation and reward, and it does not need a model of the environment [11]. The Q-learning algorithm iteratively updates the Q-values of the agent based on the rewards received for each action taken [17]. The strategy of Q-learning is to find the maximum rewards way into the future. The process commences from an initial state, where the agent selects an action based on the current Q-values. The chosen action leads to a subsequent state and yields a reward. The Q-values of the current state are updated based on subsequent state Q-values, utilizing the Q-learning algorithm [17]. To balance exploration and exploitation, the epsilon-greedy exploration strategy is employed, allowing the agent to choose random actions with a probability of epsilon, while selecting the action with the highest Q-value with a probability of 1-epsilon [6]. Decaying epsilon and decaying alpha are utilized, gradually reducing their values over time [6]. This approach enables the agent to prioritize exploration initially and shift towards exploitation and optimal decision-making as training progresses.

Q-learning in parking lot navigation optimizes the path-finding process through continual learning and trial and error [17]. By updating Q-values based on obtained rewards, the agent can make informed decisions, leading to the discovery of the quickest path to the designated parking spot. The Q-learning algorithm stores state-action Q-values in a Q-table, which is revised using the Bellman Equation Q-learning rule [5]. In Q-learning, state-action Q-values are preserved in a Q-table for the agent, undergoing revision through the Q-learning rule:

updates its Q-values based on new information.

This process is repeated until the agent reaches the parking spot, and it will keep updating the Q-table to find the optimal path to reach the parking spot in the shortest possible time.

The agent updates Q-values and selects actions until it reaches the parking spot. The Q-table undergoes continuous refinement to discover the optimal path to the parking spot in the shortest possible time. Q-learning is too simple when initializing Q-table and wastes too much time in the exploration process, causing a slow convergence speed [17].

To overcome this issue, we utilized the Double Q-learning approach to address the overestimation bias issue in traditional Q-learning algorithms. The Double Q-learning algorithm helps to mitigate this bias by maintaining two separate Q-value matrices, Q1 and Q2, and alternating their roles during the learning process [18], [19]. The formula for updating the Q-values in the Double Q-learning approach is as follows:

$$Q1(s, a) = (1 - \alpha) * Q1(s, a) + \alpha * (r + \gamma * \max(Q2(s', a')))$$
$$Q2(s, a) = (1 - \alpha) * Q2(s, a) + \alpha * (r + \gamma * \max(Q1(s', a')))$$

Q1 (s, a) represents the Q-value in the Q1 matrix for state (s) and action (a). Q2 (s, a) represents the Q-value in the Q2 matrix for state (s) and action (a). max (Q2(s', a')) represents the maximum Q-value from the Q2 matrix for the next state (s') and all possible actions (a'). max (Q1(s', a')) represents the maximum Q-value from the Q1 matrix for the next state (s') and all possible actions (a').

4. EVALUATION APPROACHES

4.1 Dijkstra Algorithm

The Dijkstra algorithm, named after its brilliant creator, Edsger W. Dijkstra, is a well-known and extensively used algorithm for finding the shortest path in a graph. [20] It is widely employed in determining the shortest path and focuses on identifying the path with the minimum distance, thereby addressing the shortest path problem. The algorithm systematically explores the graph, starting from the source node and gradually extending to other nodes. At each step, it updates the shortest distance to each visited node. specified that Dijkstra grows the shortest path tree rooted at the source node (s) by iteratively adding new nodes to the tree, increasing distances until it includes the first target node from (T), This algorithm uses a priority queue based on the smallest distance to guide the next node to visit, ensuring it prioritizes the most promising paths [21]. By continuously refining the shortest distances, the Dijkstra algorithm constructs the optimal path from the source node to all other nodes within the graph. It guarantees to find the shortest path as long as the graph contains no negative edge weights. The reliability and efficiency of the Dijkstra algorithm have made it popular for solving the shortest-path problem.

4.2 Random-Selection Algorithm

The Random-selection algorithm is a simple approach to pathfinding that mimics human decision-making in a grid environment. The agent randomly selects neighboring nodes without considering specific information about the target destination or grid structure. This approach replicates spontaneity but often generates suboptimal paths compared to more sophisticated algorithms. While not the most efficient for finding the shortest path, the Random-selection algorithm is a valuable baseline for evaluating and contrasting advanced algorithms. It provides insights into the benefits and shortcomings of different pathfinding strategies.

4.3 Implications of Using RL to Find Optimal Path

Using RL for parking lot navigation can bring numerous benefits, such as increased efficiency and improved functionality of the parking system, which positively impacts the environment and society. The system can quickly adjust to changes in parking lot dynamics and find the best route in real-time, reducing fuel consumption and carbon emissions and enhancing user experience. It is crucial to consider the ethical implications of using RL- based systems and thoroughly test and validate the agent in a controlled environment before deployment to ensure safety, reliability, and scalability.

It reflects on the ethical considerations when introducing RL systems, particularly in practical settings. Deploying an RL system for parking purposes may lead to unintended consequences like unfairly favoring some users, which could result in bias and discrimination. It could also result in job loss for parking employees, such as attendants. Therefore, it is important to consider potential consequences seriously and balance the

benefits and drawbacks before implementing them in real-world situations. An RL-powered system's deployment in the real world might be a difficult task requiring a lot of resources and knowledge. Before making the agent available to the public, it must be extensively tested and validated in a secure setting. When deploying the agent, factors such as safety, reliability, and the ability to deal with rising demands must be considered.

4.4 Analyzing Results

RL techniques offer potential outcomes for identifying the optimal path within a parking lot, considering constraints like handicap spots, electric vehicle charging spots, and specific vehicle types. Integrating RL enhances parking lot efficiency and organization. However, outcomes depend on parking lot design, training data, and algorithm execution. This section comprehensively examines the effectiveness of the Q-Learning algorithm, Dijkstra's algorithm, and the Random-selection algorithm for the shortest path problem. We implemented each method and conducted trials s on a top-of-the-line computer with an Intel Core i7 12th generation 12650H processor, an Nvidia RTX 3070 graphics card with 8GB of VRAM, and 16GB of DDR5 RAM clocked at 4800MHz. The trials were facilitated by the Anaconda distribution's Spyder IDE, enabling the execution of sophisticated machine-learning algorithms. The outcomes of these tests gave important information about how each algorithm behaved, as well as about its advantages and disadvantages and future applications. We will thoroughly present and go through these findings in the following parts. This chapter's figures, tables, and charts have been created to offer visual and quantitative support for our discussions and conclusions.

We conducted thorough testing to assess each algorithm's performance, robustness, and scalability, establishing a solid foundation for our study. Our research evaluated Time Complexity, Path Length, and Scalability as metrics for a comprehensive comparison. While time complexity was considered for assessing algorithmic performance, our primary focus was evaluating effectiveness, accuracy, and convergence. The path length metric provided insights into the algorithms' efficiency in finding shorter and optimal paths.

Scalability was evaluated by testing different input sizes and computing loads, ensuring stable performance. These metrics offered valuable quantitative measures to evaluate algorithm efficiency, effectiveness, and scalability. The results of our experiments provide essential information about each algorithm's behavior, advantages, disadvantages, and potential applications. Detailed discussions and analyses of these findings, supported by figures, tables, and charts, enhance our understanding of the research outcomes.

5. METRICS

5.1 Scalability and Time Complexity

The experiment results revealed a clear hierarchy in scalability and time complexity among the algorithms. The Q-Learning algorithm consistently outperformed its competitors, exhibiting adaptability and efficiency in more extensive and complex environments. In a 17x17 environment, Q-Learning was completed in 0.00093 seconds, while Dijkstra's algorithm took 0.000023 seconds, and the Random-selection algorithm took 0.00133 seconds. In a larger 27x27 environment, Q-Learning averaged around 0.0097 seconds, surpassing Dijkstra's algorithm, which took about 0.000034 seconds. The Random-selection algorithm had a completion time of 0.00188 seconds. These findings highlight Q-Learning's scalability and stability, making it the preferred choice for pathfinding in larger environments. While Dijkstra's algorithm had lower time complexity, it showed limitations in scalability and adaptability. Thus, the Q-Learning algorithm is a versatile and robust solution for pathfinding, especially in larger environments.

5.2 Path Length

This section explores the path lengths generated by different pathfinding algorithms operating within a grid environment, namely Q-Learning, Dijkstra, and Random-selection. The table in the Comparative Analysis summarizes crucial details, including the destination, the employed algorithm, the resulting path, and the corresponding number of steps required to reach the destination. Analyzing the number of steps for each path offers valuable insights into the efficiency and effectiveness of these algorithms in identifying the shortest path. A lower step count generally indicates a more optimal and concise path, thus highlighting the algorithm's proficiency in finding efficient paths. The "Path Length" section is an informative companion to the Comparison Table in the Comparative Analysis section, allowing for a holistic evaluation of the performance of the algorithms.

It enables us to assess not only their execution time but also the quality and length of the paths they generate.

By carefully examining the information in the "Path Length" and "Comparative Analysis" sections, we understand how the Q-Learning, Dijkstra, and Random-selection algorithms measure up against one another. This knowledge is vital for assessing their ability to solve pathfinding problems across grids of varying sizes effectively. Our investigation reveals that while Dijkstra's algorithm demonstrates a notable advantage in terms of time complexity and scalability while considering path length, Q-Learning exhibits exceptional performance levels, surpassing all examined algorithms. However, Q-Learning may not be as efficient in speed as other algorithms. On the other hand, the random-selection strategy often produces suboptimal solutions, resulting in longer and less efficient paths.

5.3 Comparative Analysis of Shortest Path Algorithms on a Grid-Based Environment

This section comprehensively analyzes three shortest-path algorithms—Q-learning, Dijkstra's algorithm, and random selection—within a grid-based environment. The 27x27 grid consists of 27 nodes, and our main objective is to evaluate their effectiveness in finding the shortest path between a source and a destination node. Q-learning, a RL algorithm, excels in adapting and learning from the environment, making it suitable for challenging path-finding problems. Dijkstra's algorithm is commonly used for finding the shortest path in road networks or traffic, while the Random-selection algorithm serves as a benchmark, randomly exploring the grid.

This analysis provides a detailed assessment of the algorithm capabilities, enabling a nuanced understanding of their performance. By closely examining the Table results, we can identify the benefits and drawbacks of each algorithm. The goal is to find the shortest path within the 27x27 grid, evaluating the algorithms based on step reduction and path length.

The analysis also considers computational efficiency, providing insights into execution times and complexity. We consider vital metrics such as finding the shortest path and computational efficiency to assess their performance. By comparing the outcomes, we gain valuable insights into the strengths and limitations of each algorithm in the grid-based environment. These findings contribute to advancing path-finding algorithms, benefiting transportation, robotics, and logistics fields. This section aims to determine the most effective algorithm with practical implementation potential, improving decision-making processes and optimizing path planning in various scenarios.

Table(1):-Table Of Comparison

Destination	Algorithm	Number of Steps	Path	Execution Time (17x17)	Execution Time (27x27)
4	Q-Learning	2	[1, 2, 4]	0.00093	0.00097
	Dijkstra	3	[1, 8, 13, 4]	0.000023	0.000034
	Random-Selection	3	[1, 3, 6, 4]	0.00188	0.00133
7	Q-Learning	3	[1, 3, 6, 7]	0.00093	0.00097
	Dijkstra	6	[1, 8, 13, 4, 18, 23, 7]	0.000023	0.000034
	Random-Selection	5	[1, 2, 5, 18, 23, 7]	0.00188	0.00133
18	Q-Learning	3	[1, 2, 5, 18]	0.00093	0.00097
	Dijkstra	4	[1, 8, 13, 4, 18]	0.000023	0.000034
	Random-Selection	6	[1, 3, 12, 17, 4, 5, 18]	0.00188	0.00133
25	Q-Learning	4	[1, 3, 4, 20, 25]	0.00093	0.00097
	Dijkstra	5	[1, 8, 13, 4, 20, 25]	0.000023	0.000034
	Random-Selection	7	[1, 2, 13, 4, 22, 27, 7, 25]	0.00188	0.00133
22	Q-Learning	3	[1, 3, 4, 22]	0.00093	0.00097
	Dijkstra	4	[1, 8, 13, 4, 22]	0.000023	0.000034
	Random-Selection	6	[1, 2, 13, 4, 6, 27, 22]	0.00188	0.00133
15	Q-Learning	2	[1, 10, 15]	0.00093	0.00097
	Dijkstra	2	[1, 10, 15]	0.000023	0.000034
	Random-Selection	4	[1, 3, 6, 4, 15]	0.00188	0.00133
27	Q-Learning	4	[1, 12, 3, 6, 27]	0.00093	0.00097
	Dijkstra	5	[1, 8, 13, 4, 22, 27]	0.000023	0.000034
	Random-Selection	5	[1, 2, 5, 4, 6, 27]	0.00188	0.00133
23	Q-Learning	4	[1, 2, 4, 18, 23]	0.00093	0.00097
	Dijkstra	5	[1, 8, 13, 4, 18, 23]	0.000023	0.000034
	Random-Selection	5	[1, 2, 5, 23, 4, 23]	0.00188	0.00133

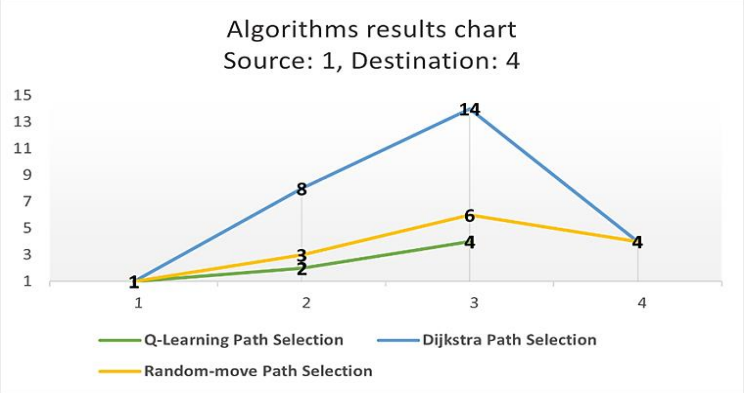


Fig. 2. For destination node 4, the Q-learning algorithm selected the path [1, 2, 4] with two steps, the shortest path among the three algorithms. Dijkstra's algorithm chose the way [1, 8, 14, 4] with three stages, slightly worse than Q-learning. The random-move algorithm selected [1, 3, 6, 4] with three steps, the same length as Dijkstra's path but with different intermediate nodes.

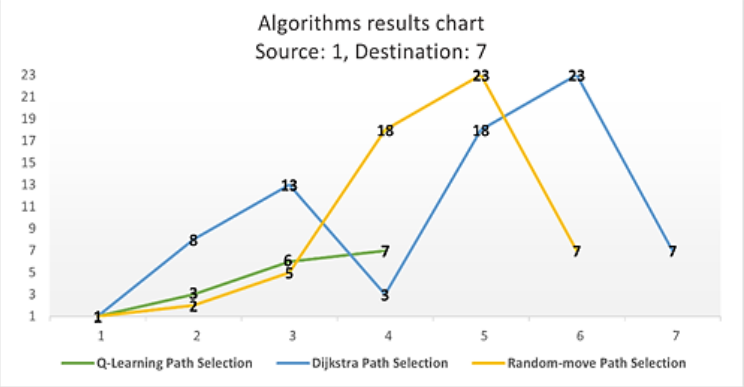


Fig. 3. For destination node 7, the Q-learning algorithm selected the path [1, 3, 6, 7] with three steps, the shortest path among the three algorithms. Dijkstra's algorithm chose the way [1, 8, 13, 3, 18, 23, 7] with six steps, much longer than Q-learning's path. The random-move algorithm selected [1, 2, 5, 18, 23, 7] with five steps, which is also longer than Q-learning's path but shorter than Dijkstra.

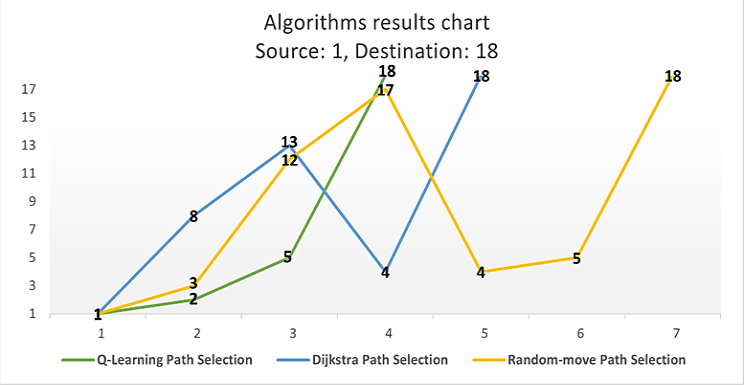


Fig. 4. For destination node 18, the Q-learning algorithm selected the path [1, 2, 5, 18] with three steps, the shortest path among the three algorithms. Dijkstra's algorithm selected the path [1, 8, 13, 4, 18] with four steps. Random-move algorithm was selected [1, 3, 12, 17, 4, 5, 18] with six steps, which was the worst result among the three algorithms.

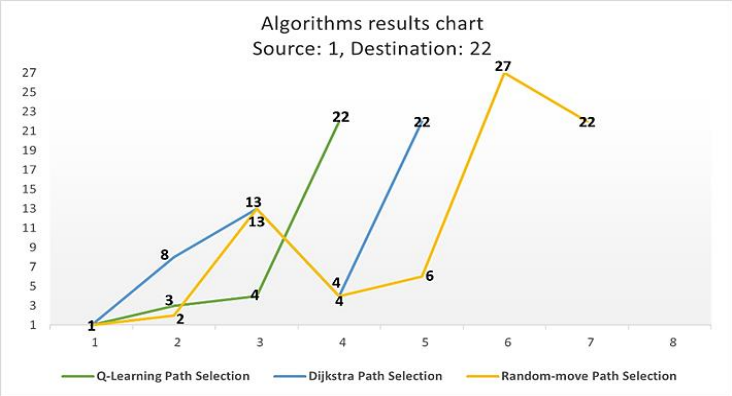


Fig. 5. For destination node 22, the Q-learning algorithm selected the path [1, 3, 4, 22] with three steps, again the shortest path. Dijkstra's algorithm selected the path [1, 8, 13, 4, 22] with four steps. The Random-move algorithm selected [1, 2, 13, 4, 6, 27, 22] with six steps.

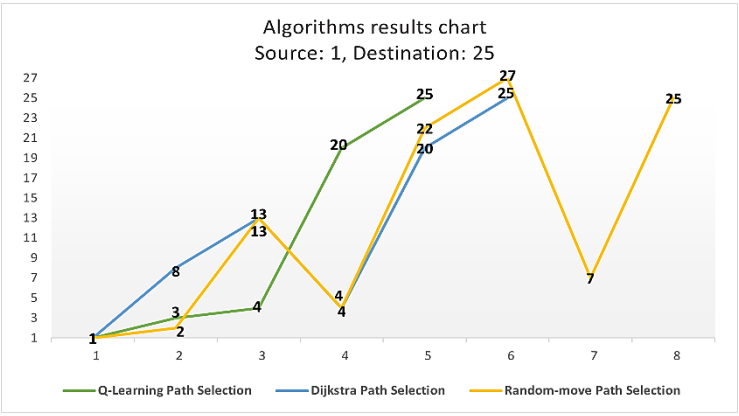


Fig. 6. For destination node 25, the Q-learning algorithm selected the shortest path in four steps [1, 3, 4, 20, 25]. In five steps, Dijkstra's algorithm picked the path [1, 8, 13, 4, 20, 25]. The Random-move method chose the path [1, 2, 13, 4, 20, 27, 7, 25] in seven steps, causing the lowest results of the three algorithms.

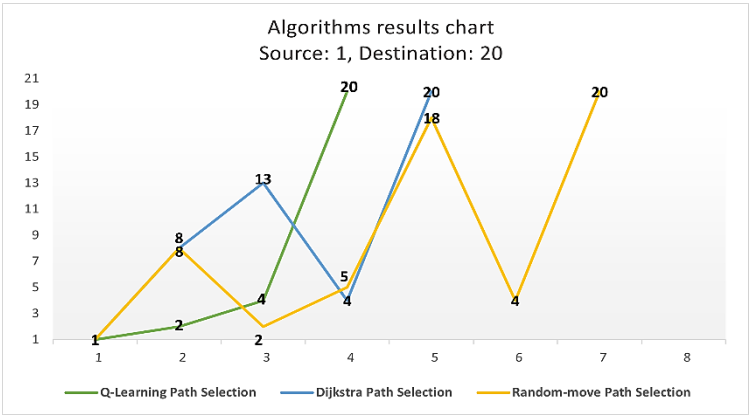
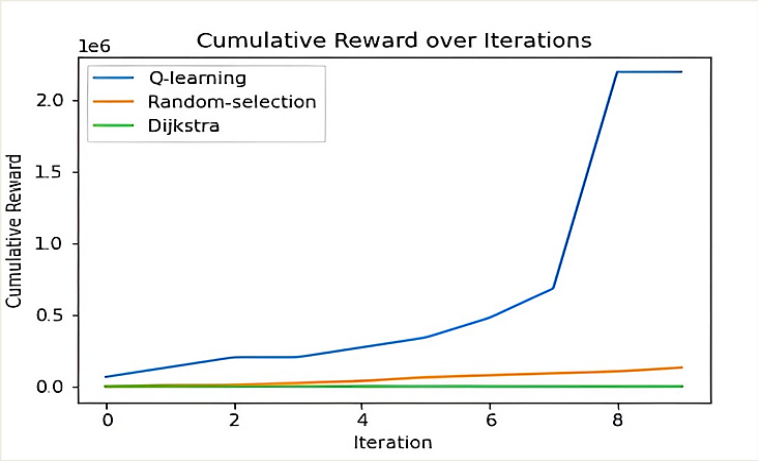


Fig. 7. For destination node 20, the Q-learning algorithm selected the path [1, 2, 4, 20] with three steps, the shortest path out of the three algorithms. Dijkstra's algorithm made its decision in four steps [1, 8, 13, 5, 20], whereas the Random-move algorithm made its decision in six steps, choosing the path [1, 8, 2, 5, 18, 4, 20].

5.4 Comparative Analysis of Cumulative Rewards in Q-Learning, Dijkstra, and Random-Selection Algorithms **in this section, we will evaluate the performance of the algorithms based on their cumulative rewards. We will compare and analyze the cumulative rewards obtained from multiple runs of each algorithm to assess their effectiveness in achieving optimal rewards. The comparison of the cumulative rewards is in Fig. 6. These findings highlight the importance of considering the trade-off between path optimality and rewards maximization when selecting an algorithm for a specific task. The Q-Learning algorithm proves advantageous in scenarios where optimal paths and cumulative rewards are preferred, while Dijkstra remains suitable for situations prioritizing minimal distance. Random-Selection is more suited for tasks where exploring different paths is necessary without focusing on rewards maximization. While Dijkstra guarantees optimality in the distance, its rewards remain lower than Q-Learning. Overall, Q-Learning demonstrates the highest effectiveness in maximizing rewards, while Dijkstra excels in finding the shortest path. Random-Selection shows lower and inconsistent cumulative rewards due to its random nature.**



The comparative analysis of the Q-learning algorithm, Dijkstra's algorithm, and random-

selection across five scenarios reveal distinct performance patterns. Q-learning outperforms the other algorithms in four out of five scenarios, efficiently addressing the shortest path problem in various graph configurations. Adaptability and reward-based learning enable effective navigation and optimal path discovery, minimizing steps to reach destinations. However, the complexity of the network can influence Q-learning's efficacy, challenging its learning process and convergence toward optimal solutions. Further research is needed to investigate its applicability and scalability across different environment settings, providing insights into adaptability and limitations.

Fig. 8. The Q-Learning algorithm achieves high cumulative rewards (ranging from 2,197,570.15 to 205,823.20), indicating effective learning and optimization of action selection. It progressively refines its policy, improving navigation and rewards. In contrast, Dijkstra's

algorithm focuses on the shortest path, resulting in modest cumulative rewards (ranging from 6 to

1) that reflect minimal path length. The Random-Selection algorithm generates highly variable cumulative rewards (ranging from 132,449 to 11,901) due to random action selection.

6. CONCLUSION

Throughout this research, we investigated into the complex realm of the shortest path challenge, focusing on a comparative analysis of three key algorithms: Q-learning, Dijkstra's algorithm, and the Random-move algorithm. Through a careful calculation on a grid featuring 27 nodes and 729 edges, we shed light on the distinctions of algorithmic performance in controlled conditions. Importantly, our findings consistently highlight the effectiveness of the Q-learning algorithm in quick discovery of optimal paths. It maintains a

clear advantage, particularly in terms of step efficiency, affirming its ability to find the shortest paths. However, our exploration reaches beyond the immediate, prompting us to inquire into the scalability and robustness dimensions of Q-learning, especially considering variations in environment complexity and size.

This effort advances the discourse on the shortest path problem, elevating it with insights into the effectiveness of different algorithmic strategies. In this evolving landscape, Q-learning emerges as a promising approach, showcasing a glimpse of its broader potential in optimal path discovery. As our understanding deepens, opportunities for further exploration needed, inviting us to untangle the various capacities and applications of these algorithms. This pursuit seamlessly integrates into ongoing research, enhancing our grasp of pathfinding and its algorithmic solutions.

7. FUTURE WORK

Future research should focus on several areas. Firstly, using larger sample sizes in trials would yield more robust findings. Increasing the number of trials and observations enables a more accurate algorithm performance assessment. Evaluating algorithm scalability on a larger scale provides valuable insights into effectiveness and efficiency. Expanding the analysis to complex ecosystems with realistic elements can enhance the understanding of algorithm applicability. Exploring deep reinforcement learning algorithms shows promise for solving the shortest path problem. Considering the influence of graph characteristics such as degree distribution and clustering coefficient is crucial. Hybrid approaches combining multiple algorithms can enhance efficiency and robustness. And also, it is essential to explore Metaheuristic Optimization Algorithms. These algorithms, inspired by natural processes, offer a powerful path to enhance algorithmic efficiency and effectiveness. Integrating these algorithms into our exploration holds the potential to further increase the adaptability, scalability, and robustness of our pathfinding solutions. The investigations will contribute to developing more efficient algorithms for practical applications.

REFERENCES

Zhang, H. Chen, S. Song, and F. Hu, “Reinforcement Learning-Based Motion Planning for Automatic Parking System,” IEEE Access, vol. 8, pp. 154485–154501, 2020, doi: 10.1109/ACCESS.2020.3017770.

S. Spanò et al., “An efficient hardware implementation of reinforcement learning: The q-learning algorithm,” IEEE Access, vol. 7, pp. 186340–186351, 2019, doi: 10.1109/ACCESS.2019.2961174.

R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction Second edition,” 2018.

L. Piardi, J. Lima, A. I. Pereira, and P. Costa, “Coverage path planning optimization based on Q-learning algorithm,” in AIP Conference Proceedings, American Institute of Physics Inc., Jul. 2019. doi: 10.1063/1.5114220.

Z. Tan, T. Wang, and Y. Yu, “Mobile robot navigation method based on improved Q-learning algorithm,” in Proceedings - 2021 36th Youth Academic Annual Conference of Chinese Association of Automation, YAC 2021, Institute of Electrical and Electronics Engineers Inc., May 2021, pp. 184–188. doi: 10.1109/YAC53711.2021.9486613.

P. B. Karthik, K. Kumar, V. Fernandes, and K. Arya, “Reinforcement Learning for Altitude Hold and Path Planning in a Quadcopter,” in 2020 6th International Conference on Control, Automation and Robotics, ICCAR 2020, Institute of Electrical and Electronics Engineers Inc., Apr. 2020, pp. 463–467. doi: 10.1109/ICCAR49639.2020.9108104.

M. Khalid, N. Aslam, and L. Wang, “A Reinforcement Learning based Path Guidance Scheme for Long-range Autonomous Valet Parking in Smart Cities,” in 2020 8th International Conference on Communications and Networking, ComNet2020 - Proceedings, Institute of Electrical and Electronics Engineers Inc., Oct. 2020. doi: 10.1109/ComNet47917.2020.9306103.

X. Liu, Q. Zhou, H. Ren, and C. Sun, 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS). IEEE, 2018.

P. Zhang et al., “Reinforcement learning-based end-to-end parking for automatic parking system,” Sensors (Switzerland), vol. 19, no. 18, Sep. 2019, doi: 10.3390/s19183996.

X. Lin and R. Guo, “Path planning of unmanned surface vehicle based on improved q-learning algorithm,” in 2019 IEEE 3rd International Conference on Electronic Information Technology and Computer Engineering, EITCE 2019, Institute of Electrical and Electronics Engineers Inc., Oct. 2019, pp. 302–306. doi: 10.1109/EITCE47263.2019.9095038.

C. Wang, X. Yang, and H. Li, “Improved Q-Learning Applied to Dynamic Obstacle Avoidance and

Path Planning," IEEE Access, vol. 10, pp. 92879–92888, 2022, doi: 10.1109/ACCESS.2022.3203072.

G. Kulathunga, "A Reinforcement Learning based Path Planning Approach in 3D Environment," in Procedia Computer Science, Elsevier B.V., 2022, pp. 152–160. doi: 10.1016/j.procs.2022.10.217.

A. Ranjan, A.K. Mahadani, and T.A. Rashid, "Multi-agent Reinforcement Learning for Stock Market Strategy Analysis," in S. Gupta, I. Banerjee, and S. Bhattacharyya (Eds.), *Multi Agent Systems*, Springer Tracts in Human-Centered Computing, Springer, Singapore, 2022, ch. 9, https://doi.org/10.1007/978-981-19-0493-6_9.

X. Wang, L. Jin, and H. Wei, "The Shortest Path Planning Based on Reinforcement Learning," in Journal of Physics: Conference Series, Institute of Physics Publishing, Jul. 2020. doi: 10.1088/1742-6596/1584/1/012006.

V. M. Babu, U. V. Krishna, and S. K. Shahensha, "An autonomous path finding robot using Q-learning," in 2016 10th International Conference on Intelligent Systems and Control (ISCO), Jan. 2016, pp. 1-6. doi: 10.1109/ISCO.2016.7727034.

Y. Hu, L. Yang, and Y. Lou, "Path Planning with Q-Learning," in Journal of Physics: Conference Series, IOP Publishing Ltd, Jun. 2021. doi: 10.1088/1742-6596/1948/1/012038.

Y. Li, H. Wang, J. Fan, and Y. Geng, "A novel Q-learning algorithm based on improved whale optimization algorithm for path planning," PLoS One, vol. 17, no. 12 December, Dec. 2022, doi: 10.1371/journal.pone.0279438.

M. Dumke, "Double Q(σ) and Q (σ, λ): Unifying Reinforcement Learning Control Algorithms," Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.01569

H. Van Hasselt, "Double Q-learning," in Proceedings of the 28th International Conference on Neural Information Processing Systems, pp. 2613-2621, 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf

Jason, M. Siever, A. Valentino, K. M. Suryaningrum, and R. Yunanda, "Dijkstra's algorithm to find the nearest vaccine location," Procedia Computer Science, vol. 216, pp. 5–12, 2023, doi: 10.1016/j.procs.2022.12.105

W. Feijen and G. Schäfer, "Dijkstra's algorithm with predictions to solve the single-source many-targets shortest-path problem," Dec. 2021, [Online]. Available: http://arxiv.org/abs/2112.11927