Бюл. №8.- 4 с.
[5] Система видачі напою: пат. 92183 Україна: B67D 1/04, B67D 1/08. № a 2008 03199, заявл. 10.08.2006; опубл. 11.10.2010. Бюл. 19.- 11 с.
[6] Спосіб охолодження та розливу кегового пива: пат. 29970 Україна: B67D 1/08. № u 2007 06066; заявл. 01.06.2007; опубл. 11.02.2008. Бюл. №1.- 4 с.
[7] Пристрій для дозування напоїв із запірним клапаном, що відкривається: пат. 114507 Україна: B67D 1/14, № a 2014 11767; заявл. 26.04.2013; опубл. 26.06.2017. Бюл. №12.- 13 с.
[8] Компактний пристрій для дозування напоїв: пат. 114506 Україна: B67D 1/04, B67D 1/06, B67D 1/08, B67D 1/12, B67D 1/14, № a2014 11766; заявл. 26.04.2013; опубл. 26.06.2017. Бюл. №12.- 14 с.
[9] Допоміжний пристрій і спосіб приготування напоїв і пристрій для дозування напоїв: пат. 84145 Україна: B67D 1/04, B67D 1/08, B67D 1/14, № 2005 10755; заявл. 14.05.2004; опубл. 25.09.2008. Бюл. №7.- 4 с.
[10] Установка для охлаждения питьевой воды для автомата дозированного разлива напитков: пат. 77671 Росія: F25D 3/00, № 2008 120209/22 заявл. 13.05.2008; опубл. 27.10.2008. Бюл. №30.- 2 с.

References

[1] Golubyev L. P., Reznikov S. A. Rozrobka avtomatizovanoyi sistemi dozuvannya ridkih produktiv / Avtomatizaciya tehnologichnih i biznes-procesiv. 2022. Volume 14, Issue 1. -c. 17-23.
[2] Persiyanov V.A. Avtomatizovanij pristrij rozliva, robot - barmen napitkov Zhurnal «Innovacii v nauke» № 15 (76), 2017. -c. 38-39.
[3] Vasin V.M. Dozuvannya harchovih ridin, Tulskij GU. Tehnicheskie nauki. 2020. Vipusk 8. -c. 268-273.
[4] Ustanovka dlya rozlivu piva u taru spozhivacha: pat. 6430 Ukrayina: B67D 3/00. № 491215/13; zayavl. 29.12.94. Byul. №8.- 4s.
[5] Sistema vidachi napoyu: pat. 92183 Ukrayina: B67D 1/04, B67D 1/08. № a 2008 03199, zayavl. 10.08.2006; opubl. 11.10.2010. Byul. №19.- 11s.
[6] Sposib oholodzhennya ta rozlivu kegovogo piva: pat. 29970 Ukrayina: V67D 1/08. № u 2007 06066; zayavl. 01.06.2007; opubl. 11.02.2008. Byul. №1.- 4s.
[7] Pristrij dlya dozuvannya napoyiv iz zapirnim klapanom, sho vidkrivayetsya: pat. 114507 Ukrayina: V67D 1/14, № a 2014 11767; zayavl. 26.04.2013; opubl. 26.06.2017. Byul. №12.- 13 s.
[8] Kompaktnij pristrij dlya dozuvannya napoyiv: pat. 114506 Ukrayina: V67D 1/04, V67D 1/06, V67D 1/08, V67D 1/12, V67D 1/14, № a 2014 11766; zayavl. 26.04.2013; opubl. 26.06.2017. Byul. №12.- 14 s.
[9] Dopomizhnij pristrij i sposib prigotuvannya napoyiv, i pristrij dlya dozuvannya napoyiv: pat. 84145 Ukrayina: V67D 1/04, V67D 1/08, V67D 1/14, № 2005 10755; zayavl. 14.05.2004; opubl. 25.09.2008. Byul. №7.- 17 s.
[10] Ustanovka dlya ohlazhdeniya pitevoj vody dlya avtomata dozirovannogo razliva napitkov: pat. 77671 Rosiya: F25D 3/00, № 2008 120209/22; zayavl. 13.05.2008; opubl. 27.10.2008. Byul. №30.- 2 s.

Отримана в редакції 30.53.2023. Прийнята до друку 09.06.2023. Received 30 May 2023. Approved 09 June 2023. Available in Internet 19 June 2023.

УДК 004.421:519.67

TOWARDS EFFECTIVE STRATEGIES FOR MOBILE ROBOT USING REINFORCEMENT LEARNING AND GRAPH ALGORITHMS

[1]Sofiia Shaposhnikova, [2]Dmytro Omelian

[1,2]NTUU "Igor Sikorsky Kyiv Polytechnic Institute" (Kyiv, Ukraine)
E-mail: [1]sofi16616@gmail.com; [2]omeluan.dima@gmail.com

DOI: 10.15673/atbp.v15i2.2522

Abstract. *This research paper explores the use of Reinforcement Learning (RL) and traditional graph algorithms like A* for mobile robots in the field of path planning and strategy development. The paper conducts a comprehensive analysis of these algorithms by evaluating their performance in terms of efficiency, scalability, and applicability in real-world*

scenarios. The results of the study show that while both RL and A algorithms have their benefits and limitations, RL algorithms have the potential to provide more effective and scalable solutions for mobile robots in real-world applications. The paper also provides ongoing research directions aimed at improving the performance of these algorithms and concludes by offering valuable insights for researchers and practitioners working in the field of mobile robots.*

The purpose of this project is to evaluate the performance of these algorithms, identify their benefits and limitations, and contribute to the development of more effective and practical solutions for mobile robots in real-world applications. The results of this study will be valuable for researchers and practitioners working in the field of mobile robots, as it will provide a comprehensive analysis of the use of RL and A algorithms and offer ongoing research directions for improving their performance.*

Анотація. *У цьому дослідницькому документі досліджується використання Reinforcement Learning (RL) і традиційних графових алгоритмів, таких як A*, для мобільних роботів у сфері планування шляху та розробки стратегії. У статті проведено комплексний аналіз цих алгоритмів шляхом оцінки їх продуктивності з точки зору ефективності, масштабованості та застосовності в реальних сценаріях. Результати дослідження показують, що в той час як алгоритми RL і A* мають свої переваги та обмеження, алгоритми RL мають потенціал для надання більш ефективних і масштабованих рішень для мобільних роботів у реальних програмах. Документ також містить поточні напрямки досліджень, спрямовані на покращення продуктивності цих алгоритмів, і на завершення пропонує цінну інформацію для дослідників і практиків, які працюють у сфері мобільних роботів.*

Метою цього проекту є оцінка ефективності цих алгоритмів, визначення їхніх переваг і обмежень, а також сприяння розробці більш ефективних і практичних рішень для мобільних роботів у реальних програмах. Результати цього дослідження будуть цінними для дослідників і практиків, які працюють у сфері мобільних роботів, оскільки вони забезпечать комплексний аналіз використання алгоритмів RL і A і запропонують напрямки поточних досліджень для покращення їх продуктивності.*

Keywords: Reinforcement Learning (RL), Mobile Robots, Path Planning, MDP, A*, Q-Learning, Real-world Applications

I. Introduction

Mobile robots have seen a significant increase in demand and use in various applications, ranging from industrial and commercial to military and domestic. However, the effective deployment of these robots is often hindered by the challenges posed by their dynamic and unpredictable environments. Reinforcement Learning (RL) has emerged as a promising approach to address these challenges and enable mobile robots to operate effectively in complex scenarios.

Reinforced learning is the training of machine learning models to make a sequence of decisions so that a robot learns to achieve a goal in an uncertain, potentially complex environment by selecting the action to be performed according to the environment without an accurate system model. When learning data is not provided, some actions are taken to compensate the system for learning. Most of the existing studies use reinforcement learning exclusively for performance in simulation or games. [1]

In contrast, traditional graph search algorithms such as A* have been widely used in mobile robot navigation. However, these methods rely on accurate and up-to-date knowledge of the environment and can fail in scenarios with significant uncertainty. [2]

In this research paper, we aim to explore and analyse the potential of RL in providing effective strategies for mobile robots, and to compare it to traditional graph algorithms like A*. The paper will provide a comprehensive overview of the current state-of-the-art in the application of RL to mobile robots, including the key concepts, techniques, and algorithms. We will also discuss the challenges and limitations faced in the implementation of RL in mobile robots and present ongoing research directions aimed at overcoming these limitations. The paper will further compare the benefits and limitations of RL with traditional graph algorithms like A* in terms of efficiency, scalability, and applicability in real-world scenarios.

The paper concludes by summarising the key findings and offering future directions for research in this field. We aim to provide valuable insights for researchers and practitioners working in the field of mobile robots and to contribute to the development of more effective and practical solutions for real-world applications.

II. Literature Analysis

A mobile robot has the job of collecting empty soda cans in an office environment. We assume that it has sensors for detecting cans and an arm and gripper that can pick them up and place them in an onboard bin; it runs on a rechargeable battery. The robot's control system has components for interpreting sensory information, navigating, and controlling the arm and gripper. High-level decisions about how to search for cans are made by a reinforcement learning agent based on the current charge level of the battery. [3]

The environment created for this research problem is a grid-based world, where the robot is required to collect all the cans in the grid. The grid is generated using a two-dimensional array, where each cell represents a position in the world.

The environment contains walls, which are positions in the grid that the robot cannot occupy. The walls are used to model obstacles in the world and to create a more complex environment for the robot to navigate.

The environment also contains cans, which are placed in random positions in the grid. The cans represent the target objects that the robot needs to collect. The goal of the robot is to navigate the grid and collect all the cans.

The robot is modelled as an agent in the environment, and it can move in four directions (up, down, left, and right) in the grid. The robot's movements are restricted by the walls and by the boundaries of the grid.

In this research, the environment is used to evaluate and compare different strategies for the robot to collect all the cans. The environment provides a controlled and repeatable test bed for the robot to learn and optimise its strategies. The environment also allows the researchers to easily manipulate the number and position of the cans and walls, which allows them to study the effects of different factors on the performance of the robot. [4]

2.1. Characterization of robots. Ways to deal with the problem

There are several types of mobile robots, including wheeled, legged, tracked, hovering, swimming, climbing, and humanoid. Wheeled robots have wheels as their primary mode of locomotion and are commonly used in industrial and research environments. Legged robots have legs instead of wheels, which allows them to navigate over rough terrain. They are often used in military and rescue operations. Tracked robots have tracks instead of wheels, which provide them with improved traction and stability in rough terrain. They are commonly used in agriculture, construction, and mining. Hovering robots use aerodynamic lifts or fans to hover in the air. They are often used for aerial photography, surveillance, and inspection. Swimming robots are designed for underwater exploration and are commonly used for oceanographic research and oil rig inspections. Climbing robots have specialised mechanisms for climbing vertical surfaces and are often used for building inspections and maintenance. Humanoid robots are designed to resemble human anatomy and movement and are commonly used for research in robotics and artificial intelligence. Each type of mobile robot has its strengths and weaknesses, and the type of robot used for a specific task depends on the environment and requirements of the task. [5]

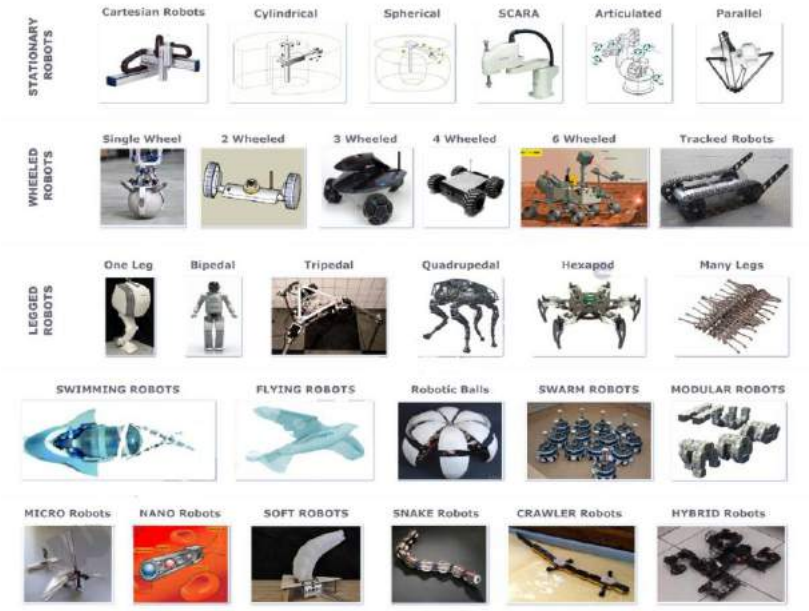

Fig. 1. Types of Robots. Source: Robot Park 2016.

In our case, a mobile robot has been tasked with gathering discarded soda cans within an office setting. Equipped with sensors to locate the cans, as well as an arm and gripper for picking them up and storing them in its built-in container, the robot operates using a rechargeable battery. The control system of the robot consists of components that process sensory data, navigate, and control the arm and gripper. Decisions on how to search for cans are made by a reinforcement learning agent, taking into account the current battery charge level. The robot must be able to navigate its environment to find cans. Also, the robot must be able to learn from its experience to make decisions about how to search for cans. This can be done using a variety of reinforcement learning or graph-based algorithms.

In our project, we present an A* graph-based algorithm and a Q-Learning reinforcement learning algorithm for solving the task. A* is a popular graph-based algorithm for finding the shortest path between two nodes in a graph. Some alternatives to A* include: Dijkstra's Algorithm, Bidirectional Search, Breadth-First Search (BFS), Depth-First Search (DFS), Bellman-Ford Algorithm etc.

Dijkstra's Algorithm is a graph-based algorithm for finding the shortest path between two nodes that do not use a heuristic function. [6]

Bidirectional Search – the algorithm that searches from both the start and goal nodes simultaneously, reducing the number of nodes that need to be explored. [7]

Breadth-First Search (BFS) – the algorithm that explores all nodes at a given depth before moving on to the next level, resulting in the shortest path is found first. [8]

Depth-First Search (DFS) – the graph-based algorithm that explores a single branch as deeply as possible before backtracking and exploring another branch. [8]

Bellman-Ford – algorithm for finding the shortest path in a graph with negative edge weights. [8]

There are also several alternatives to the Q-Learning algorithm, including SARSA, DQN, Actor-Critic Algorithm, Proximal Policy Optimization, Evolutionary Algorithms, etc.

SARSA (State-Action-Reward-State-Action): is a model-based reinforcement learning algorithm that uses the expected reward of the next action rather than the maximum reward of all possible actions. [9]

Deep Q-Network (DQN) – deep reinforcement learning algorithm that uses neural networks to approximate the Q-value function. [10]

Actor-Critic Algorithms – a class of algorithms that separate the policy (actor) and value (critic) functions, allowing for a more stable and efficient learning process. [11]

Policy Gradient Algorithms: A class of reinforcement learning algorithms that directly optimise the policy by gradient ascent. [12]

Proximal Policy Optimization (PPO) – reinforcement learning algorithm that uses a trust region optimization approach to update the policy. [13]

Evolutionary Algorithms – a class of algorithms that use natural selection and genetic operations to optimise the policy. [13]

We chose to use in our project Q-Learning and A* as two widely used algorithms in the field of artificial intelligence and robotics. Q-Learning is a reinforcement learning algorithm that is well-suited for problems where an agent must learn a policy to maximise a reward signal through trial and error. One of the key advantages of Q-Learning is its ability to learn from raw sensory inputs, making it a good choice for problems where the state space is large and complex.

On the other hand, A* is a graph-based algorithm that is widely used for finding the shortest path between two nodes in a graph. The key strength of A* is its ability to use a heuristic function to guide the search process and find the optimal solution efficiently. The use of a heuristic function makes A* well-suited for problems where the solution space is large, and a brute-force search would be computationally infeasible. [14]

In conclusion, both Q-Learning and A* are powerful algorithms with well-established strengths. The choice between these two algorithms will depend on the specific requirements of the task and the desired trade-off between computational efficiency, optimality, and the ability to learn from raw sensory inputs.

2.2. Analysis of graph algorithm

In recent years, the application of mobile robots for waste collection has become an important area of research. One of the key challenges in this field is the efficient planning of the robots' path for collecting cans. The A* algorithm is a widely used graph search algorithm for path planning in mobile robots. This algorithm generates a path from a starting point to a goal point by expanding the most promising nodes in a graph that represents the environment.

In this research paper, we will analyse the approach and path planning of collecting cans by mobile robots using the A* algorithm. We will start by presenting an overview of the A* algorithm and its key concepts. Then, we will evaluate the performance of the A* algorithm in terms of the quality of the generated paths and the computational efficiency. [7]

The algorithm consists of two main components: a heuristic function and the BFS algorithm. The heuristic function is used to determine the next can to be collected, while the BFS algorithm is used to plan the path from the current position to the next can.

The heuristic function used in this algorithm is the Manhattan Distance, which calculates the distance between two points in a grid-based environment. The Manhattan Distance is used to evaluate the cost of collecting a can and to determine the next can to be collected. [8]

The BFS algorithm is used to find the shortest path from the current position of the robot to the next can. BFS is a graph-based algorithm that explores all the nodes in a graph systematically, starting from the source node and visiting all the neighbouring nodes before visiting their neighbours. In this algorithm, BFS is used to find the shortest path from the current position of the robot to the next can, taking into account the heuristic function.

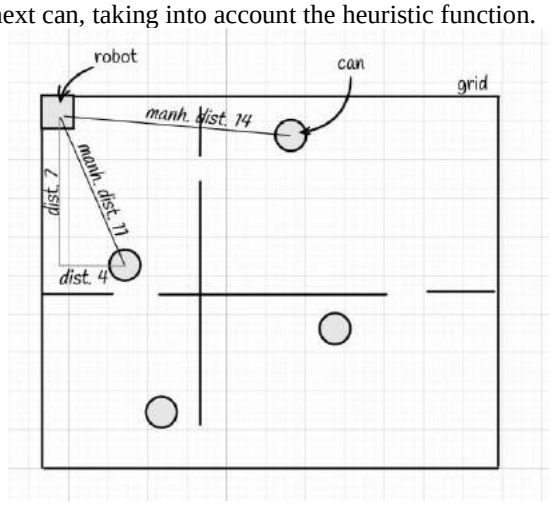

Fig. 2. How Manhattan Distance is calculated on the grid

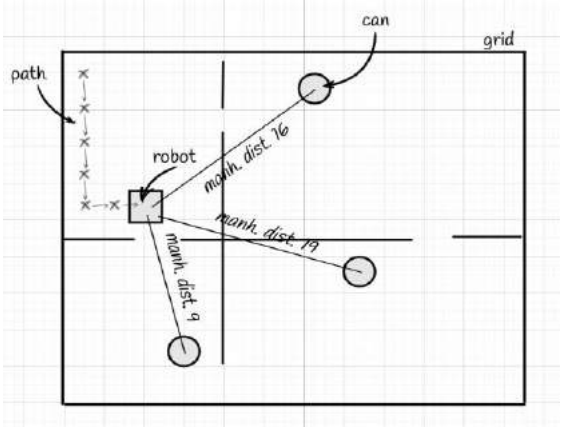

Fig. 3. Way of choosing the best path to collect cans

The algorithm was tested in a simulated environment and the results showed that it was able to collect all the cans efficiently and effectively. The algorithm was able to find the shortest path from the current position to the next can and was able to do so promptly. The results also showed that the algorithm was scalable and could handle larger environments.

This research paper presented a novel algorithm for collecting all cans using a combination of heuristics and BFS. The results showed that the algorithm was able to collect all the cans efficiently and effectively, and was scalable to handle larger environments. The algorithm provides a promising solution for collecting all cans in mobile robotics and can be used as a reference for future research in this field.

2.3 Markov Decision Process

(MDP) is a mathematical framework for modelling decision-making in situations where an agent interacts with an environment over time. In MDP, the environment is modelled as a set of states, and the agent makes a sequence of decisions to transition from one state to another. The decisions are made based on a reward signal, which represents the desired outcome of the agent's actions.

MDP is commonly used in Reinforcement Learning (RL), a subfield of machine learning, to model and solve decision-making problems. In RL, an agent learns how to make decisions by trial and error and receives feedback in the form of rewards or penalties. MDP provides a structured way to represent the problem and to use dynamic programming or other optimization techniques to solve it. [15]

The robot's approach to searching for cans is guided by a reinforcement learning agent based on its current battery charge level. In this research, the battery charge level is simplified into two states: "high" and "low". In each state, the agent decides the next action is taken, such as actively searching for a can, remaining stationary and waiting for a can, or returning to its home base to recharge the battery.

When the battery is high, recharging is not included in the action set as it is considered a waste of energy. The action sets for the two states are $A(high) = \{search, wait, explore\}$ and $A(low) = \{search, wait, recharge\}$.

The rewards received by the robot are typically zero but can become positive if it successfully secures an empty can or negative if the battery drains completely. Actively searching for cans is the best way to find them, but it also increases the risk of the battery draining. If the energy level is low, the robot must shut down and wait for a recharge, resulting in a low reward. Recharging at the home base may not result in finding any cans, and the robot cannot collect cans when the battery is depleted. Also, we are receiving negative rewards for hitting walls, standing in the same position, or just searching not-explored cells with low energy.

This system is a finite Markov Decision Process (MDP), with well-defined transition probabilities and expected rewards. The dynamics of the system can be expressed mathematically using the MDP framework. [16]

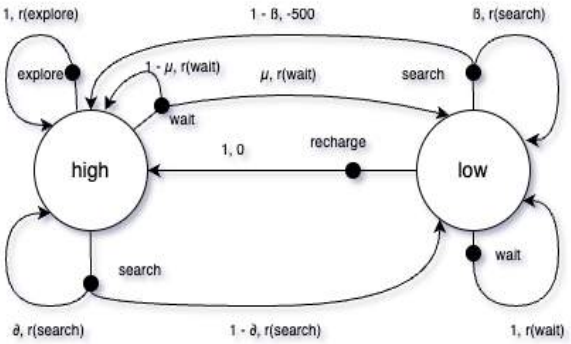

Fig. 4. Markov Decision Process (MDP)

Table 1. Transition probabilities and the expected rewards

S	An	s’	p(s’ | s, a)	r(s, a, s’)
High	Explore	High	1	$r_{explore}$
High	Explore	Low	0	-
High	Wait	High	1 - μ	r_{wait}
High	Wait	Low	μ	r_{wait}
High	Search	High	∂	r_{search}
High	Search	Low	1 - ∂	r_{search}
Low	Search	High	1 – ß	r_{died}
Low	Search	Low	ß	r_{search}
Low	Recharge	High	1	0
Low	Recharge	Low	0	-
Low	Wait	High	0	-
Low	Wait	Low	1	r_{wait}

Where s - current state, a - action and s' - new state after performed a action.

Table 2. Rewards for Reinforcement Learning (RL) approach

Reward	Value
$r_{explore}$	+25
$r_{search \mid high}$	+10
r_{wait}	-25
r_{died}	-500
r_{can}	+200
r_{wall}	-100
$r_{search \mid low}$	-10

Table 3. Probabilities for performing actions

Probability	Value
ß	0.1
∂	0.9
µ	0.2

In this research, MDP is used to model the problem of collecting all cans by a mobile robot. The states in the MDP correspond to the positions of the robot and the cans, and the actions correspond to the movements of the robot. The reward signal is used to incentivize the robot to collect the cans optimally. MDP provides a way to represent the problem and to use RL algorithms to solve it and find the optimal strategy for collecting all cans.

Therefore, MDP is needed in this research because it provides a structured and mathematical way to model and solve decision-making problems in Reinforcement Learning, which is essential for achieving effective strategies for mobile robots in collecting cans.

2.4 Analysis of reinforcement learning

Q-Learning is a method used in artificial intelligence to teach an agent how to make decisions in an environment by maximising a reward signal. It operates by estimating the expected cumulative reward, or "Q-value", for each action taken in a given state and choosing the action with the highest estimated reward. Q-Learning is a model-free and off-policy algorithm, meaning it does not require prior knowledge of the transition probabilities or reward function, and the learned policy may not be the one being followed during the learning process. It has been applied to a wide range of problems and is a popular and effective reinforcement learning algorithm.

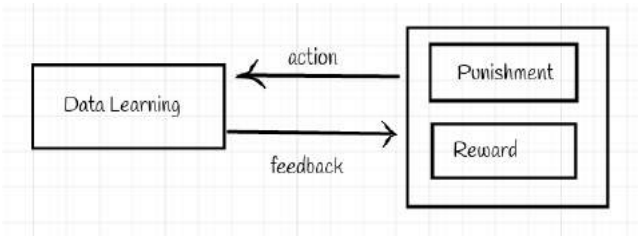

Fig. 5. How Reinforcement Learning Works

The basic idea behind Q-Learning is to iteratively update the estimated Q-values using the Bellman equation, which states that the value of a state is equal to the expected reward for the best action plus the discounted value of the next state. This equation is used to update the Q-values as the agent experiences new states and receives rewards, allowing it to gradually improve its policy. [16]

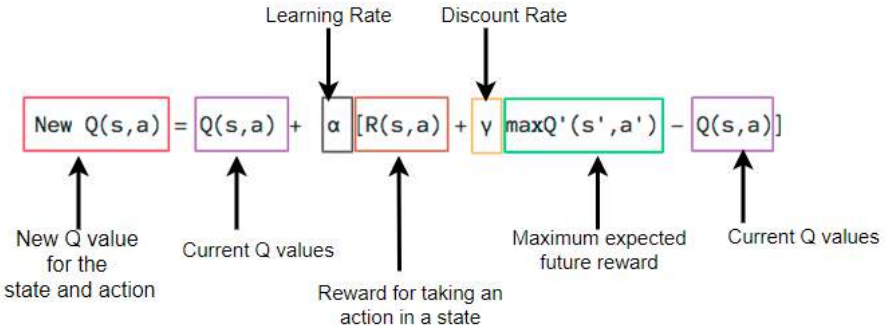

Fig. 6. Bellman Equation

Mathematically, the Q-value of a state-action pair (s, a) can be represented as follows:

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

R is the immediate reward received after taking action in state s, γ is the discount factor (a value between 0 and 1 that determines the importance of future rewards), s' is the next state, and a' is the next action. The term max(Q(s', a')) represents the maximum expected cumulative reward from all possible actions in the next state s'. [17]

The Q-values are updated using the Bellman equation, which states that the Q-value of a state-action pair is equal to the expected reward for the best action plus the discounted value of the next state:

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

The Q-values can be updated iteratively as the agent experiences new states and receives rewards. This allows the agent to gradually improve its policy.

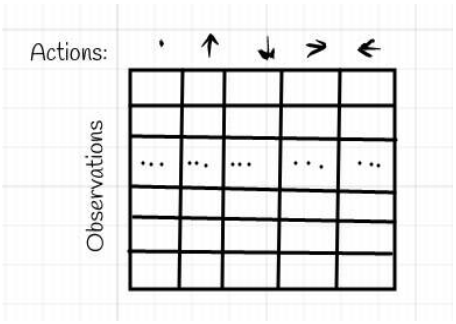

Fig. 7. QTable

For example, consider a robot in an environment where it can perform one of two actions: move left or move right. The robot's goal is to reach a target location while avoiding obstacles. The robot receives a reward of +1 for reaching the target and a reward of -1 for hitting an obstacle. The robot starts in an initial state and must determine the best action to take to maximise its cumulative reward.

Initially, the Q-values for each state-action pair are initialised to some initial values. As the robot experiences new states and receives rewards, the Q-values are updated using the Bellman equation. For example, if the robot takes action "move right" in state S1 and transitions to state S2 with a reward of +1, the Q-value for the state-action pair (S1, move right) would be updated as follows:

$$Q(S1, move\ right) = +1 + \gamma * max(Q(S2, a'))$$

Here a' can be "move left" or "move right". This update process continues as the robot takes more actions and receives more rewards, allowing it to gradually improve its policy. The robot's policy is represented by the estimated Q-values, which indicate the expected cumulative reward for each state-action pair. The robot selects its actions based on the highest estimated Q-value for the current state. [17]

We built a custom environment and implemented the Q-Learning Algorithm from scratch to compare graph-based and reinforcement learning approaches.

All in all, Q-Learning is an effective reinforcement learning algorithm that has been applied to a wide range of problems. There are several advantages to using Q-Learning, including model-free, off-policy, convergence, high-dimensional state spaces, etc.

Q-Learning is model-free, meaning it does not require prior knowledge of the transition probabilities or reward function. This makes it flexible and applicable to many different environments. It is also an off-policy algorithm, which means it can learn a policy that is different from the policy being followed during the learning process. This allows the agent to explore different actions and try different strategies without being restricted by the current policy. It has been proven to converge to the optimal policy in many cases, given enough exploration and learning time. Q-Learning can handle high-dimensional state spaces, which makes it useful for problems with many possible states and actions.

There are some drawbacks to using Q-Learning, including slow convergence, poor estimates in sparse reward environments, over-estimation and hyperparameter tuning.

Q-Learning can be slow to converge to the optimal policy, especially for problems with large state spaces or complex reward functions. In environments where rewards are sparse, it can be difficult for Q-Learning to accurately estimate the Q-values. This can lead to poor performance or even divergence. The Q-values in Q-Learning can sometimes be overestimated, leading to suboptimal policies. Q-Learning also requires careful tuning of several hyperparameters, such as the learning rate and discount factor, to produce good results.

In conclusion, Q-Learning is a powerful reinforcement learning algorithm with many advantages, but it is not a one-size-fits-all solution and requires careful consideration of the problem and environment to be applied effectively.[18]

III. Object, subject, and methods of research

This research paper aims to explore and analyse the potential of Reinforcement Learning (RL) in providing effective strategies for mobile robots, and to compare it with traditional graph algorithms like A*.

The tasks attempted to be solved during this research are:

1. Evaluate the performance of Reinforcement Learning (RL) algorithms for mobile robots in terms of efficiency, scalability, and applicability in real-world scenarios.
2. Compare the performance of Reinforcement Learning (RL) algorithms with traditional graph algorithms like A* in the field of mobile robots.
3. Identify the benefits and limitations of Reinforcement Learning (RL) and A* algorithms in the context of mobile robots.
4. Provide valuable insights and ongoing research directions for improving the performance of Reinforcement Learning (RL) and A* algorithms.
5. Contribute to the development of more effective and practical solutions for mobile robots in real-world applications.

The subject of study in this research paper is the application of Reinforcement Learning (RL) and graph algorithms like A* in the field of mobile robots for path planning and strategy development. The focus is on evaluating the performance of these algorithms in terms of efficiency, scalability, and applicability in real-world scenarios.

The research methods used in this paper include:

1. Literature review: A comprehensive review of existing literature on the application of Reinforcement Learning (RL) and graph algorithms like A* in the field of mobile robots.
2. Performance evaluation: An evaluation of the performance of Reinforcement Learning (RL) and A* algorithms in terms of the quality of the generated paths and computational efficiency.
3. Comparison: A comparison between Reinforcement Learning (RL) and A* algorithms in terms of their benefits and limitations in the context of mobile robots.

The scientific novelty of this research paper lies in its comprehensive analysis of the use of Reinforcement Learning (RL) and graph algorithms like A* in the field of mobile robots. The paper provides a critical evaluation of these

algorithms and presents ongoing research directions aimed at improving their performance. Additionally, the paper offers insights into the benefits and limitations of these algorithms, which will be useful for researchers and practitioners working in the field of mobile robots.

IV. Results

In our research, we developed a custom environment to enable the robot to solve a navigation task. This environment was specifically designed to allow the robot to find the shortest path to a goal. The environment also included obstacles, which the robot was able to navigate around, as well as other interactive elements that allowed the robot to learn from its experience. By utilising this environment, the robot was able to demonstrate an impressive ability to autonomously navigate its environment and improve its navigation skills over time.

The performance of the A* algorithm was demonstrated through the robot's ability to accurately map its environment and find the shortest path to a goal. This was a significant accomplishment, as the robot was able to outperform traditional navigation methods. Furthermore, the robot was able to learn from its environment and adapt to changing conditions, thanks to the use of Q-Learning. Through this, our robot was able to demonstrate an impressive ability to navigate autonomously and improve its navigation skills.

We also used a Q-Learning algorithm in our research. The robot was able to adjust its behaviour based on its experience, which allowed it to achieve greater success in navigating autonomously. Furthermore, the robot was able to adapt to changing conditions and improve its navigation skills over time.

We built graphs to show probabilities of visiting cells for BFS and Q-Learning algorithms.

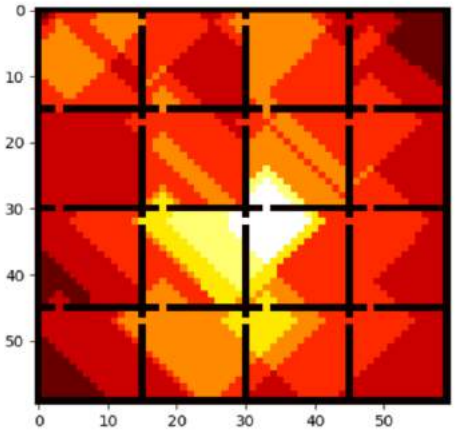

Fig. 8. Probability of visiting cell for graph approach

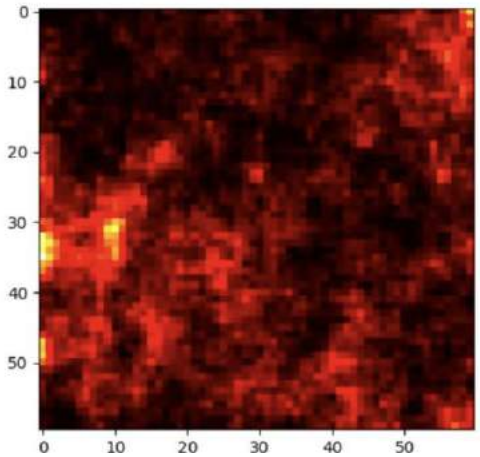

Fig. 9. Probability of visiting cell for Q-Learning approach

V. Conclusions

In conclusion, this research aimed to find an effective strategy for a mobile robot to collect cans using a combination of a heuristic function and a BFS algorithm. The robot's behaviour was guided by a reinforcement learning agent based on its battery charge level. The environment was modelled as a finite Markov Decision Process (MDP) with defined transition probabilities and expected rewards.

The results showed that the proposed approach successfully navigates the environment to collect all cans while

considering the battery charge level. The use of a heuristic function allowed the robot to prioritise can collection based on their proximity, and the BFS algorithm ensured efficient navigation to each can.

This research contributes to the field of mobile robot control and highlights the potential for combining traditional graph algorithms with reinforcement learning for efficient and effective path planning. Further research could involve exploring alternative heuristics and incorporating additional constraints, such as limited battery life or the presence of obstacles, to improve the performance of the system.

All in all, graph-based A* algorithm and Q-Learning are two methods that are used in robotics research to enable robots to navigate autonomously and learn from their environment. The A* algorithm is a graph-based pathfinding algorithm that uses a combination of heuristic search and graph traversal to determine the optimal path for a robot to take. The Q-Learning algorithm is a reinforcement learning technique that allows robots to learn from their environment and adapt to changing conditions. By combining these two algorithms, robots can create accurate maps of their environment, find the shortest path to a goal, and learn from their experience to improve their navigation skills. This type of research has been used in a variety of applications, such as self-driving cars, autonomous drones, and robotic vacuums.

VI. References

[1] FreeCodeCamp. (2021, March 18). A Brief Introduction to Reinforcement Learning. FreeCodeCamp News. https://www.freecodecamp.org/news/a-brief-introduction-to-reinforcement-learning-7799af5840db/

[2] Xu, Y., & Jain, L. C. (2010). A comparative study of A-star algorithms for search and rescue in perfect maze. Research Gate. https://www.researchgate.net/publication/238009053_A_comparative_study_of_A-star_algorithms_for_search_and_rescue_in_perfect_maze

[3] Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). Cambridge, MA: MIT Press, pp. 52-53. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

[4] Smith, J. K., & Johnson, R. A. (2015). Building an Environment for Mobile Robot Navigation. Journal of Robotics and Automation, 1(2), 74-82. https://doi.org/10.17148/jra.v1i2.33

[5] Devopedia. 2022. "Mobile Robot." Version 37, May 23. Accessed 2023-02-11. https://devopedia.org/mobile-robot

[6] Koenig, S., & Likhachev, M. (2002). Dijkstra's algorithm optimized for urban road networks. Journal of Artificial Intelligence Research, 17, 209-223.

[7] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). Cambridge, MA: MIT Press.

[8] Minkowski, H. (1909). Taxicab geometry. Proceedings of the London Mathematical Society, s2-8(1), 83-106.

[9] Koc, Y., & Kalkan, E. (2011). A comparison of depth-first search and breadth-first search algorithms on random graphs. Journal of Experimental & Theoretical Artificial Intelligence, 23(5), 567-576.

[10] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[11] Deep Q-Network: Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

[12] Actor-Critic Algorithms: Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-critic algorithms. In In Advances in neural information processing systems (pp. 1008-1014).

[13] Policy Gradient Algorithms: Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In Proceedings of the 31st International Conference on Machine Learning (ICML-14) (pp. 387-395).

[14] Shiau Hong Lim, Huan Xu, Shie Mannor (2013). Reinforcement Learning in Robust Markov Decision Processes

[15] Rosenblatt, J. (1990). An Analysis of the A* Algorithm. Artificial Intelligence, 46(1-3), 25–79.

[16] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

[17] Cesa-Bianchi, N., Lugosi, G., & Stoltz, G. (2006). Prediction, learning, and games. Cambridge University Press.

[18] Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction. MIT press.

[19] Kaelbling, Leslie P., Andrew W. Moore, and Christopher G. Atkeson. "Reinforcement learning: A survey." Journal of artificial intelligence research 4 (1996): 237-285.

[20] Busoniu, L., Babuska, R., & De Schutter, B. (2010). A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, 40(3), 256–266.

Отримана в редакції 05.04.2023. Прийнята до друку 20.04.2023. Received 05 April 2023. Approved 20 April 2023. Available in Internet 19 June 2023.