JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Comparative Analysis of Robotic Path Planning Techniques Using Reinforcement Learning

[1]Durga B
[1]PG Scholar
[1]Department of Artificial Intelligence,
[1]Reva University, Bengaluru, India

Abstract : This review paper examines advanced learning-based strategies for path planning in autonomous robots, focusing on their application in unknown and dynamic terrains. The study delves into five distinct approaches: Q-learning, Bayesian learning, Deep reinforcement learning (DRL), Fuzzy Q-learning, and neural networks combined with hierarchical reinforcement learning (HRL). Each approach is analyzed for its ability to navigate complex environments, handle uncertainties, and optimize path planning efficiency. Q-learning is explored for its iterative learning capabilities in unpredictable terrains, while Bayesian learning is assessed for its probabilistic decision-making in scenarios with limited information. The paper also reviews studies from DRL applied to grid path planning, highlighting its potential for handling large state spaces. Fuzzy Q-learning is discussed for its effectiveness in managing imprecision, and the integration of neural networks with HRL is examined for its ability to decompose complex tasks into manageable sub-goals.

The comparative analysis provides insights into the strengths and limitations of each technique, with a focus on their adaptability, computational demands, and real-world applicability. The paper concludes with a discussion of future directions, emphasizing the potential for hybrid approaches and the need for further research to enhance scalability and robustness in robotic path planning.

Index Terms - Q Learning, Deep Reinforcement Learning, Hierarchical Reinforcement Learning, Bayesian Learning, Path Planning, Fuzzy Learning

I. INTRODUCTION

Mobile Robots are a class of robots that mainly focus on movement from point A to B based on several input criteria and state space constraints. This process is mainly divided into three modules which include information perception, path planning and motion control [1]. Advanced systems help in good path-planning scenarios and in turn, can reduce the wear and tear of the robot. Continuous development of path planning can lead to more efficient systems within the mobile robots and in turn can take advantage of reduced lead time and quick start and goal scenarios. According to Mohanty et al [2] path planning can be divided into global and local environments. In the case of global path planning, the robot is completely aware of the environment and can reach the desired destination within the stipulated time. However, in the case of local path planning, the robot is unclear of the environment and needs continuous inputs, thereby enabling the robot to reach the destination based on predefined inputs. These inputs can be either open-loop or closed-loop. This can provide a quick solution, however, the disadvantage is that globally it may not lead to the desired goal or sometimes may be unreachable. There is no hard and fast rule to ensure the utilization of relevant algorithms to sustain local or global path planning. The only process that needs to be followed is that utilizing an appropriate algorithm for each method can lead to better solutions. Since the 1950's there have been several algorithms proposed by authors worldwide and one such algorithm is A* which is even today being utilized by Google for its Maps platform [3]. Since then several algorithms have been in place and have enabled the growth of the path-planning scenario. From a fundamental point of view, simple learning algorithms will suffice the need for shortest-path planning, but on a global level with a complex environment, reinforcement learning plays a vital role in reaching a solution (Machine Learning Workflow) [4].

In many cases, supervised learning is an ideal choice for fixed environments, with change in scenario robot needs to gather real-time data [5] Reinforcement Learning (RL) also belongs to the same class of learning [6].

II Q LEARNING (QL)

Q-learning is a classic form of RL that has proved its worthiness in recent years and is popular as it provides a model-free Temporal Difference (TD) [7,8]. In this process, the best action to be taken in a given state to maximize the reward is worked on. The major key concepts include :

- **Action**: The decision made by the agent in each state.
- **State**: The current situation the agent is in.
- **Reward**: Feedback received after taking an action, guiding learning.
- **Q-Values**: Estimates of future rewards for each state-action pair.
- Environment: Where the agent operates.
- **Policy**: Strategy derived from Q-values for selecting the best action.
- **Q-Table**: A table storing Q-values for all state-action pairs.
- **Bellman Equation**: Formula used to update Q-values based on new experiences.
- **Cumulative Reward**: The total reward accumulated over time.
- **Residual**: The difference between updated and previous Q-values.
- **One Episode**: A full sequence of actions from start to goal or terminal state.
- **Horizon**: The length or range of time over which rewards are evaluated.

Methodology:

The agent (robot) interacts with the environment and updates its Q-table based on the rewards it receives. The Bellman Equation is used to update Q-values, which is fundamental in MDPs and FMDPs for evaluating and improving the policy. The Q-Learning algorithm is implemented as it does not require a predefined path and can adapt to new environments. The robot's actions include moving forward, turning left, or turning right based on its sensory inputs.

The Following pseudocode for Q learning Algorithm is used to find the optimal action-selection policy for a given finite Markov Decision Process (FMDP)., which refers to MDPs with a finite number of states and actions., The experiment involves a robot with a defined and finite environment, making it suitable for FMDP. This makes the problem tractable and easier to handle computationally.

```
Initialize  Q(s,  a)  arbitrarily  for  all
states s and actions a
Set the learning rate α (0 < α ≤ 1)
Set the discount factor γ (0 ≤ γ < 1)
Set the exploration rate ε (0 ≤ ε ≤ 1)

Repeat (for each episode):
    Initialize the starting state s

    Repeat (for each step in the episode):
        With probability ε, select a random
action a (explore)
        Otherwise, select the action a that
maximizes Q(s, a) (exploit)
```

Fig 1.0 Pseudocode

- **States (s)**: Represent different situations the agent can be in.
- **Actions (a)**: The possible decisions the agent can make in each state.
- **Rewards (r)**: Feedback the agent receives after taking an action.
- **Q(s, a)**: A function that estimates the future rewards for taking action *a* in state *s*.
- **α (Alpha)**: Learning Rate - Determines the weight given to new information when updating Q-values.
- **γ (Gamma)**: Discount Factor - Reduces the value of future rewards compared to immediate rewards.
- **ε (Epsilon)**: Exploration Rate - Probability of choosing a random action to explore new options.

The algorithm iteratively updates the Q-values (which represent expected future rewards) to improve the agent's policy, allowing it to make better decisions over time.

In work carried out by Prases K. Mohanty et al [9] on Q Learning analysis, the authors used a Microcontroller STM3F4 at 168MHZ, FPU and DSP core along with ROS platform, initialized with zero values for 8 states and 3 actions, guided a robot in simulations and physical experiments using an E-Puck2 mobile robot. The robot, equipped with LIDAR, odometry, and IMU sensors, navigated toward a target while avoiding obstacles, with sensor readings from 0°, 30°, and 330° influencing the Q-Table updates over 200 episodes. The results demonstrated robust obstacle avoidance and path optimization in both simulated and real-world environments. Future work aims to address dynamic obstacles for enhanced adaptability.

III. D-Q LEARNING

DQN (Deep Q-Learning) combines Q-Learning with deep neural networks. replaces the Q-table with a neural network to approximate Q-values, allowing it to handle larger and more complex state spaces. It uses a replay memory to store past experiences and sample minibatches for training, which helps break correlations between consecutive experiences. It separates the target network to stabilize training. This network's weights are periodically updated to match the main Q-network.

The grid environment used in this study is static and consists of traversable and non-traversable cells (obstacles). The task for the agent is to find a path from the start to the goal on this grid. The agent is trained on small grid maps, and its ability to generalize is tested on previously unseen maps.

In a paper addressed by Aleksandr Panov et [10] the challenge of pathfinding on an 8-connected grid with blocked (obstacle) and unblocked (traversable) cells. The goal is to find a path from a start to a goal location, where moves have different costs (1 for cardinal and √2 for diagonal). The agent must navigate around obstacles using a limited field of view and receive rewards based on its actions and proximity to the goal. To solve this, the paper employs Deep Q-Learning, a model-free, off-policy reinforcement learning method, which approximates the Q-function using a deep neural network with convolutional and fully connected layers. The experience replay technique is used to stabilize learning by storing and sampling past experiences.

The results demonstrate that the deep Q-Learning approach effectively learns to navigate complex environments and find paths with minimal cost. The agent successfully handles dynamic and obstructed environments, showing promising performance in simulations and real-world experiments. While the approach achieves robust pathfinding and efficient learning, the study notes that there is still room for improvement in terms of convergence and adaptability to diverse obstacle shapes.

IV. BAYESIAN LEARNING

Q-learning updates Q-values deterministically based on rewards. DQN uses deep neural networks for approximating Q-values and incorporates experience replay. **Bayesian Q-learning**, however, models Q-values as probability distributions, incorporating uncertainty and improving decision-making by considering both prior knowledge with observed data [11]. This approach is useful for making informed decisions in uncertain environments, allowing for more flexible and adaptive modeling.

The study addresses the challenge of motion planning for high-Degree-of-Freedom (DOF) robots in partially unknown or incomplete environments, where conventional methods like POMDPs (partially observable Markov decision processes) are not feasible due to discretization issues.

The **BLIND** (Bayesian Learning in the Dark) algorithm aims to produce safe trajectories for robots [12] in incomplete or partially unknown environments by integrating human feedback into the motion planning process. It combines Bayesian Inverse Reinforcement Learning (BIRL) with optimization-based motion planning. BLIND uses a task model roadmap (G) to capture past experiences and human preferences for safety, guiding a motion planner to produce collision-free trajectories. It incorporates human feedback through critiques to update the reward function used in the roadmap.

BLIND achieved the highest success rate (0.86) and the fewest human queries needed (average of 2.96) compared to the baselines, demonstrating its effectiveness in producing safe trajectories with minimal human interaction. The results show that the proposed framework can successfully guide the robot in discovering feasible and safe paths in environments with incomplete information.

Pseudocode:

1. **Initialize Roadmap:**
   - G is created from past trajectories or a motion planner.
2. **Connect Start/Goal:**
   - $G' \leftarrow \text{ConnectToRoadmap}(G, \text{start}, \text{goal})$
3. **Guidance Search:**
   - $P \leftarrow \text{GuidanceSearch}(G')$
4. **Guided Motion Planning:**
   - $T \leftarrow \text{GuidedMP}(W, P, \text{start}, \text{goal})$
5. **Human Evaluation:**
   - If the trajectory T is not accepted:
     - $(D^+, D^-) \leftarrow \text{Critique}(T)$
     - $\text{UpdateRoadmap}(P(R_{D^+}, D^-))$
6. **Return Result:**
   - If trajectory T is accepted, return T.

V. Bayesian Learning

Fuzzy logic uses fuzzy sets and rules to handle imprecise or vague information [13]. It translates qualitative information into quantitative decisions using membership functions and rule-based systems, Ideal for systems requiring human-like reasoning and dealing with ambiguous data, such as control systems and decision-making processes with uncertain inputs.

For example, instead of saying "distance is 5 meters," fuzzy logic might use "distance is close" or "distance is far," and it uses membership functions to quantify these terms.

In **Fuzzy Q-Learning**, fuzzy logic is used to interpret sensory inputs (like distance from obstacles) and convert them into fuzzy values. These values are then used in Q-Learning to update and improve the decision-making process. Instead of using precise inputs, the system uses fuzzy rules to handle uncertainties and make better decisions [14].

In an article written by Sepideh Valiollahi et al [15], he addresses the challenge of autonomous robot navigation, which involves optimizing for speed, safety, and obstacle avoidance. Combining fuzzy logic with Q-learning., allows the robot to adaptively tune its fuzzy parameters in real-time through reinforcement learning, improving its ability to navigate effectively while avoiding obstacles.

The findings show that the fuzzy Q-learning method significantly enhances robot navigation performance compared to traditional methods. It achieved a 95% probability of successfully reaching the goal and improved safety margins by 15%. The method facilitated smoother and more efficient navigation by dynamically adjusting to the environment, leading to better obstacle handling and faster goal attainment in simulated scenarios. This demonstrates the method's effectiveness in providing adaptive and robust navigation solutions.

VI. Hierarchical Reinforcement Learning

Hierarchical Reinforcement Learning (HRL) [16] is an approach that addresses complex decision-making problems by decomposing them into simpler, more manageable sub-tasks. In HRL, the primary idea is to create a hierarchy of policies, where higher-level policies set goals or sub-tasks for lower-level policies to achieve.

HRL is used to tackle complex environments where a single-level policy may struggle due to the large state and action spaces. By breaking down the problem into a hierarchy of tasks, HRL aims to make learning more efficient and scalable.

Path Planning Algorithms

- **Q-Learning**: A tabular method that updates the value of state-action pairs based on rewards and is suitable for environments with discrete states and actions.
- **Deep Q-Network (DQN)**: Uses neural networks to approximate the Q-values, improving the handling of large state spaces.
- **Potential DQN (PDQN)**: Enhances DQN with artificial potential fields to accelerate learning.
- **Actor-Critic (A3C)**: Utilizes both value and policy networks to handle high variance in reward signals.
- **Deep Deterministic Policy Gradient (DDPG)**: Combines neural networks with policy gradients for continuous action spaces.
- **Double DQN (DDQN)**: Addresses overestimation bias in Q-value estimation by using two separate networks.

The path planning for a two-wheel differential mobile robot involves controlling the speeds of its two driving wheels to navigate arbitrary trajectories, including linear movement, turns, and circular paths. The robot's motion is described by a kinematic model.

- **Robot Type**: Two-wheel differential mobile robot.
- **Movement Control**: The robot controls the speed of its left and right wheels to achieve movements like linear motion, turning, and circling.

In a study done by Yu J et al [17] found that advanced algorithms like Deep Q-Network (DQN), Potential DQN (PDQN), Actor-Critic (A3C), Deep Deterministic Policy Gradient (DDPG), and Double DQN (DDQN) significantly enhance path planning performance compared to basic methods. These advanced techniques reduced planning time, improved path efficiency, and increased cumulative rewards, demonstrating superior navigation capabilities.

Advanced reinforcement learning algorithms (DDQN and DDPG) are more effective in optimizing mobile robot path planning than traditional methods. They offer better performance in complex environments, making them preferable for achieving efficient and effective navigation.

VII. Comparison

Criteria	Q-Learning	Deep Q-Learning (DQN)	Bayesian Q-Learning	Fuzzy Q-Learning	Hierarchical Reinforcement Learning (HRL)
Complexity of State Space	Small, discrete	Large, continuous	Uncertain	Imprecise	Decomposed
Adaptability	Moderate	High	High	High	Very High
Learning Efficiency	Slower	Faster	Efficient	Efficient	Efficient
Handling of Uncertainty	Limited	Complex	Excellent	Good	Moderate
Real-World Application	Simple	Complex	Uncertain	Ambiguous	Highly complex
Scalability	Limited	High	Scalable	Scalable	Highly Scalable
Convergence	Slow	Improved	Variable	Depends	Variable

VIII Conclusion

In conclusion, mobile robots' path planning has evolved significantly, with advanced reinforcement learning techniques like Q-Learning, DQN, and their variants offering robust solutions for navigating complex environments. Traditional methods, while effective in simpler scenarios, often fall short in dynamic and uncertain environments. Integrating deep learning, Bayesian methods, and fuzzy logic into path planning enhances adaptability and decision-making, enabling robots to operate efficiently in both known and unknown terrains. The continuous improvement of these algorithms, particularly in handling larger state spaces and real-time learning, marks a substantial advancement in the field. Future research should focus on refining these techniques for better convergence, adaptability, and real-world applicability, ensuring mobile robots can autonomously and reliably achieve their goals in increasingly complex environments.

References

[1] Lixing Liu, Xu Wang, Xin Yang, Hongjie Liu, Jianping Li, Pengfei Wang, Path planning techniques for mobile robots: Review and prospect, Expert Systems with Applications, Volume 227, 2023, 120254, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.120254

[2] P.K. Mohanty, A.K. Singh, A. Kumar, Path planning techniques for mobile robots: A review International Conference on Soft Computing and Pattern Recognition, Springer, Cham (2021), pp. 657-667

[3] Miyombo Ernest Miyombo, Yong-kuo Liu, Chishinga Milton Mulenga, Anthony Siamulonga, Martin Chihango Kabanda, Phillimon Shaba, Chunli Xi, Abiodun Ayodeji, Optimal path planning in a real-world radioactive environment: A comparative study of A-star and Dijkstra algorithms, Nuclear Engineering and Design, Volume 420, 2024, 113039, ISSN 0029-5493, https://doi.org/10.1016/j.nucengdes.2024.113039

[4] Zhang X., Xia S., Li X. Multi-objective particle swarm optimization with multi-mode collaboration based on reinforcement learning for path planning of unmanned air vehicles Knowl.-Based Syst., 250 (2022), Article 109075

[5] Konar I., Chakraborty G., Singh S.J., A deterministic improved Q-learning for path planning of a mobile robot IEEE Trans. Syst. Man Cybern.: Syst., 43 (5) (2013), pp. 1141-1153

[6] Džeroski S., Raedt D.L., Driessens K. Relational reinforcement learning Mach. Learn., 43 (2001), pp. 7-52

[7] Low E.S., Ong P., Low C.Y., Omar R. Modified Q-learning with distance metric and virtual target on path planning of mobile robot Expert Syst. Appl., 199 (2022), Article 117191

[8] Tan X., Zhang X.Q., Shi H.G., Reinforcement-learning-based multi-slot rail empty wagon real-time distribution J. Transport. Eng. Inf., 18 (4) (2020), pp. 53-60

[9] P. K. Mohanty, S. Saurabh, S. Yadav, Pooja and S. Kundu, "A Q-Learning Strategy for Path Planning of Robots in Unknown Terrains," 2022 1st International Conference on Sustainable Technology for Power and Energy Systems (STPES), SRINAGAR, India, 2022, pp. 1-6, doi: 10.1109/STPES54845.2022.10006505.

[10] AleksandrI.Panov, KonstantinS.Yakovlev, RomanSuvorov, Grid Path Planning with Deep Reinforcement Learning Preliminary Results, 8th annual conference on biological inspired cognitive architecture, BICA 2017, Proceedia Computer Science 123 (2018) 347-353

[11] Camilla van Geen, Raphael T. Gerraty, Hierarchical Bayesian models of reinforcement learning: Introduction and comparison to alternative methods, Journal of Mathematical Psychology, Volume 105, 2021, 102602, ISSN 0022-2496, https://doi.org/10.1016/j.jmp.2021.102602

[12] C. Quintero-Peña, C. Chamzas, V. Unhelkar, and L. E. Kavraki, Motion Planning via Bayesian Learning in the Dark, InICRA 2021: Workshop on Machine Learning for Motion Planning, 2021,

[13] Majed Alateeq, Witold Pedrycz, Logic-oriented fuzzy neural networks: A survey, Expert Systems with Applications, Volume 257, 2024, 125120, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2024.125120

[14] Thanh Trung Nguyen, Minh Hai Vu, Thi Ha Ly Dinh, Thanh Hung Nguyen, Phi Le Nguyen, Kien Nguyen, FQ-SAT: A fuzzy Q-learning-based MPQUIC scheduler for data transmission optimization, Computer Communications,
Volumes 226–227, 2024, 107924, ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2024.107924

[15] S. Valiollahi, R. Ghaderi and A. Ebrahimzadeh, "A fuzzy Q-learning approach to navigation of an autonomous robot," The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012), Shiraz, Iran, 2012, pp. 520-525, doi: 10.1109/AISP.2012.6313802.

[16] Pieter Verbeke, Tom Verguts, Reinforcement learning and meta-decision-making, Current Opinion in Behavioral Sciences, Volume 57, 2024, 101374, ISSN 2352-1546, https://doi.org/10.1016/j.cobeha.2024.101374

[17] Yu J, Su Y and Liao Y (2020) The Path Planning of Mobile Robot by Neural Networks and Hierarchical Reinforcement Learning. Front. Neurorobot. 14:63. doi: 10.3389/fnbot.2020.00063