SN Computer Science (2023) 4:447
https://doi.org/10.1007/s42979-023-01876-0

ORIGINAL RESEARCH

A Novel Algorithm for Optimal Trajectory Generation Using Q Learning

Manoj Kumar[1] · Devendra Kumar Mishra[2] · Vijay Bhaskar Semwal[3]

Received: 8 February 2023 / Accepted: 10 May 2023

Abstract

The current study proposes a unique algorithm for shortest trajectory creation based on q learning. Major issues towards grid world problem are environment generalization. In Q learning to learn without prior knowledge of the system is based on trial-and-error interaction using reward and penalty. Every decision contains in the form of look-up table. The decision-making system train the agent over a series of episodes. In this research paper, we present novel algorithms for optimal trajectory analysis based on state action using pairs. Performance comparisons with various learning algorithms in the context of trajectory efficiency verses number of episodes and accuracy prediction between number of episodes shows that our proposed algorithm is better than Q Learning. This approach can be used in autonomous sectors, computer vision, route optimization along with IoT (internet of things) and distributed systems.

Keywords Q-learning · Shortest trajectory · Decision-making system · Environment · Grid world

Introduction

Agent learning, also known as reinforcement learning, is a type of machine learning that involves training an agent to make decisions based on feedback from its environment. In this paradigm, the agent interacts with an environment, receives feedback in the form of rewards or penalties for its actions, and learns to optimize its behavior to maximize the cumulative reward over time. The agent typically uses a trial-and-error approach to learn from its experiences, adjusting its behavior based on the feedback it receives from the environment. Over time, the agent learns to recognize patterns in the feedback and develops strategies to maximize its reward. Agent learning has been used to train machines to play games, control robots, and navigate complex environments. It has also been used in fields such as finance, healthcare, and transportation to optimize decision-making processes and improve outcomes [1–3].

An agent that uses reinforcement learning (RL) can perceive their environment and learn the best course of action to reach their goal. Any action taken by the agent will result in a feedback signal from the environment known as a reward. RL is learning to link situations to actions to optimize a numerical reward signal. The learner [1–5] in RL is a decision-making agent that acts in the world and is rewarded (or punished) for doing when attempting to solve a problem. After several trial-and-error runs, it ought to discover the ideal course of action: the combination of steps that maximizes the overall reward [6–9].

The interactions between an agent and its surroundings are the subject of artificial intelligence research. Multiple instances that an agent can see, affect through action constitutes an environment. By continuously acquiring new information and skills, the agent can adapt to changes in its environment and make better decisions, leading to improved performance and success in achieving its goals. With scalar evaluative feedback, RL refers to a group of learning algorithms that aims to approximatively solve random sequential decision-making tasks [10, 11]. Multi-agent environment has practical complexity where designing an optimal solution for

This article is part of the topical collection "Machine Intelligence and Smart Systems" guest edited by Manish Gupta and Shikha Agrawal.

✉ Manoj Kumar
mannu175@yahoo.com

Devendra Kumar Mishra
dkmishra@gwa.amity.edu

Vijay Bhaskar Semwal
vsemwal@manit.ac.in

[1] Amity University-Gwalior Campus, Gwalior, India

[2] Amity University-Gwalior, Gwalior, M.P., India

[3] MANIT, Bhopal, M.P., India

Published online: 14 June 2023

reward functions becomes quite situational. Then, through trial-and-error interactions with its environment, the computer learns how to accomplish that objective. At its core, reinforcement learning involves an agent learning to make decisions through trial and error, aiming to achieve the highest possible reward in a given environment by adapting its behavior based on the feedback it receives [12–16].

We describe an RL system as a five-tuple consisting of S, A, RF, and VF, where S is a set of environmental states, A is a set of actions the agent can take. The other parts are, in that order, policy, reward function (RF), and value function (VF) [17–20]. Robot navigation in uncontrolled environment is still a complex task. Localization, mapping, and optimization in open environments for agent has drawn continuous effort in trajectory generation and path planning problems [31].

A policy maps an environmental state and an action the agent is supposed to do. In other words, the agent must act per the policy. Finding the best course of action is the aim of learning. A policy is typically stochastic [21].

A reward function maps an environment's state or state-action pair to a numeric value referred to as a reward signal. This is a sign that the state or state-action pair is desirable. Given that rewards play a vital part in RL systems, attention must be given to make sure that the reward function represents the main purpose of the system rather than attaining a sub-goal. A return is a topic that is related in a lot of ways [22–25]. First, when calculating the present return, it prioritizes recent benefits over future gains. Second, it enables the use of a single definition of return for both ongoing tasks with a lengthy life span and episodic tasks that naturally divide into subsequences with a final state (such as playing chess). An RL agent's objective is to act [26]. The anticipated return an RL agent can get is defined by the value function of a specific policy. Two value functions, in particular are intriguing [27]. The expected return under the policy, starting from taking action a in state s, is defined as: $Q(s,a) = E[Rt \mid st = s, at = a]$ where Q(s,a) is the action-value function for taking action a in state s, Rt is the return, and $E[Rt \mid st = s, at = a]$ is the expected return starting from state s and taking action a. The notation "Ist = s,at = a" is equivalent to writing "st = s, at = a" and represents the indicator function that takes the value 1 when the condition is true and 0 otherwise. [28]. A method that constantly takes advantage is greed. The exploration entails acting in a manner distinct from that of greed. Investigating potential alternatives to greedy action is the goal of exploration. Exploitation and exploration are both done via the e-greedy technique. It takes the greedy action, i.e., exploits, with probability 1−, where it is a tiny positive number, and with probability e, it chooses an action at random [29].

Rewords, policy, and Environment are the core four components of any learning system; an environment is represented by a set of states. A decision-maker who perceives and chooses an action for the system is a learning agent. Currently, temporal difference network (TDN) techniques are used to evaluate RL for control problems like the grid world. These strategies solve the issue of forecasting time-delayed rewards by computing future rewards. They serve as a more accurate predictor of future performance than sampled (commutative) incentives [30]. The issue of how an autonomous agent that observes and acts in its Environment may learn to select the best course of action to accomplish its objectives is addressed by reinforcement learning (RL). When acting in an environment with a huge search area, the RL offers a general framework and a number of ways to help it behave better because the learning process can take a very long time to converge. In contrast to supervised learning techniques, reinforcement learning tasks do not give the learner access to the best action outcome. As a result, the learner must experiment with changing its policy.

Agent learning can be used in conjunction with humanoid robots to enable them to learn and adapt to their environment. Humanoid robots are robots that are designed to resemble human beings in terms of their physical appearance and abilities. Using agent learning techniques, humanoid robots can learn to interact with humans and their surroundings more naturally and effectively. Humanoids walking is driven from human walk called gait analysis. Gait analysis is combination of stance and swing phases which is further divided into eight subphases.

For example, an agent learning humanoid robot could be trained to recognize and respond to human speech, gestures, and emotions, as well as to perform tasks such as picking up objects, walking, and navigating through obstacles. The robot could also learn to adapt its behavior based on feedback from its environment, such as adjusting its walking speed or posture in response to changes in terrain.

Agent learning can also help humanoid robots to improve their performance over time, as they accumulate experience and learn from their successes and failures. This can be especially important in applications where the robot is required to perform complex or dynamic tasks, such as healthcare or manufacturing. Overall, the combination of agent learning and humanoid robotics has the potential to enable robots to become more autonomous, adaptable, and capable of interacting with humans in a natural and intuitive way.

Humanoid as agent [33–35] has many core concepts which makes humanoid agent complex in nature. Trajectory generation and path finding in controlled and dynamic environments has major challenges such as understanding of kinematics and kinetics, center of mass (COM), ZMP (zero moment point), push recovery and many more.

Implementation of humanoid walking using Q-learning for shortest path finding has following steps:

1. Define the environment: define the environment where the humanoid is walking. This can be a 2D or 3D grid with obstacles, walls, and the humanoid's starting position and target position.
2. Define the actions: define the set of actions the humanoid can take in each state. For walking, the humanoid can move in any of the four directions (up, down, left, right). Each action should have a corresponding reward.
3. Define the Q table: define the Q table, which is a matrix that contains the *Q*-values for each state-action pair. Initialize the *Q*-values to zero.
4. Train the model using Q-learning: train the model using Q-learning algorithm, which is a reinforcement learning algorithm. In each iteration, the humanoid selects an action based on the *Q*-values and explores the environment. The *Q*-values are updated using the Bellman equation.
5. Execute the model: after training, execute the model and let the humanoid walk to the target position. The path taken will be the shortest path found by the Q-learning algorithm.

In this paper we have focused on optimal trajectory generation based on episodic improvement. We will show agent learning improves as number of episodes increases. Our proposed learning algorithms results has been achieved using MATLAB simulator.

Future trends of agent learning [37] is developing multi-agent environment which is inspired from many agents in acting environment working together to optimize network goral. Real example of this learning can be working of employee in any organization to achieve day to day goals in system and communicating among themselves for obtaining internal rewards (penalty may be also there). Thus, they help each other towards achieving global reward for system (Fig. 1). Multi-agent reinforcement learning (MARL) has its own complexity and research challenges while dealing with different sectors like autonomous, computer vision, UAVs (unmanned vehicle), path planning and trajectory generations, social networks etc.

Motivation

Optimal trajectory generation is a critical problem in robotics and autonomous systems. It involves planning a path for a robot or autonomous vehicle that is both safe and efficient, while considering various environmental constraints such as obstacles, terrain, and dynamic changes. Traditional methods for trajectory generation can be time-consuming and may not always produce the best results. However, with the advent of ML-RL techniques, new opportunities have emerged for more efficient and effective trajectory generation. Humanoid as agent performs daily task which has repetitive nature. They can be used to pick and drop objects in limited environments with more accuracy and less time.

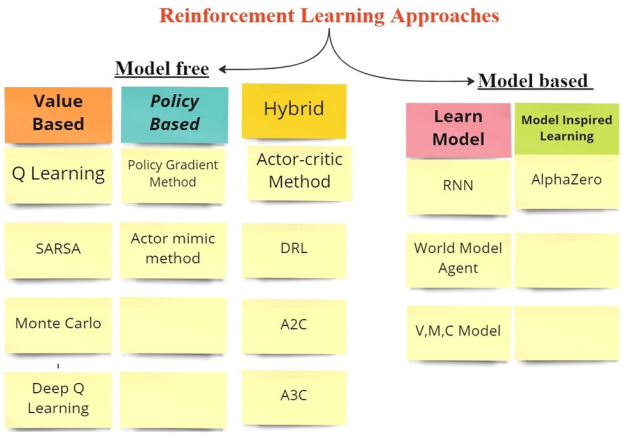

Fig. 1 Classification of different reinforcement learning approaches in machine learning [37, 38]. *DRL* deep reinforcement learning), *RNN* recurrent neural network), *V, M* world model, *C* controller model, *A2C* advantage actor-critic, *A3C* asynchronous advantage actor-critic). Empty space shows future unknown algorithms

Section one describes the introduction about agent learning, humanoid as agent and reinforcement Q learning. Section two contains details about related work done in this direction. Proposed methodology and learning algorithm has been discussed in section three while section four gives result discussion information. Section five will conclude the whole work and will tell the future scope of it.

Related Works

In recent years several works have been done in the field of trajectory generation using machine learning. Agent learning in grid environment have been performed and tested. This research review focuses on different aspect of path following performance matrix and different critical issues such as kinematics constraints, dynamic modeling, uncertainty, multi-objective optimization and real-time performance. We have done review assessment of last two decades published papers from different reputed journals and conferences. The authors [2] aim to develop a new algorithm that can learn effectively in a dynamic and uncertain environment. The proposed algorithm combines the Q-learning and SARSA algorithms to create a swarm-based learning method that can learn from the experience of individual agents and the group as a whole. The algorithm is designed to adapt to changes in the environment and make decisions based on the

behavior of other agents in the swarm. The authors validate the effectiveness of the proposed algorithm through simulations of a pursuit-evasion game. The results show that the swarm-based algorithm outperforms the traditional SARSA method and is more robust to changes in the environment.

Authors [3] have applied new dynamic neural network to reinforcement learning approach. They have tried to minimize the effect of look-up Q table and raised point of different issues in table. Future scope and challenges has been discussed. Quinn et al. [4] have applied reinforcement learning to spider as new technology. Paper throws light on RL applications in vertical search space which is inspired from Spider. Spider as agent who is searching from initial website(start) to final page of required (target) website has been analyzed and search engine called Spider_Engine has been developed. Comparative results between Spider_Engine and Nutch (web crawler) showed that Spider_engine search more documents in less time than Nutch. It was a good learning example from nature creature.

This paper [5] proposes a novel approach for optimizing time warp simulation using reinforcement learning techniques. The authors demonstrate the effectiveness of their approach through experiments on a benchmark simulation model. Overall, this paper presents an interesting application of reinforcement learning in simulation optimization. Santos-Pata et al. [6] have proposed a unique learning mechanism inspired from rodent's path finding in dynamic environments using vicarious trial and error (VTE) method. Authors have used space representation and mental travel using place and grid cells which helps further to move the agent (rodents) in desired location using reward and penalty mechanism. This study is helpful in biomimicking the nature species.

This study [7] proposes learning architecture which is a combination of imitating learning and reinforcement learning. System generates internal reward which help to achieve fast learning than error and trial mechanism. Simulated experiments of different learning scheme under two major categories showed that integrative methods are good enough to accelerate learning than other methods namely simple, shaping, and supervised learning based on fixed learning rate, discount factor and reward points. Thus, paper exercised integration learning.

This article [8] focuses on challenges in multi-agent environment where partial observation, actions, positions, intercommunication, and reward distribution is major concerns. Authors have proposed RL framework which is inspired from real life task distribution. The approach is evaluated in several scenarios and shows promising results in terms of increasing coordination and reducing conflict. However, the paper lacks a more in-depth analysis of the limitations and potential drawbacks of the proposed approach. Efroni et al. [9] have analyzed different RL algo based on trajectory feedback reward value. Authors explained that calculating state-action reward at each visited point like in case of self-driving car is quite complex and costly. Authors have work on calculating hybrid optimum trajectory feedback reward using least square error (LSE). Thompson sampling (TS with RL) is worth usable in this work. This work is worth useful to understand TF (trajectory feedback) in practical scenario.

Authors of paper [10] have applied Q learning in wireless network area where multiple radio access network (MRAN) as agents communicate to others to remove blockage and improve network performance. Proposed algorithm called DNSA (dynamic network self-optimization) has shown promising results in terms of less complexity and better network revenue. This kind of learning scheme can be motivated to apply in radar technology and aviation industry.

Sivamayil et al. in paper [11] have widely covered different application areas where RL has been used. This is one of the fundamental papers which covers all aspects for researchers and academies. Paper contains applications of RL in gaming, robotics, autonomous systems, natural language processing (NLP), marketing, finance, energy conservation and many more.

Ulusoy et al. in this [12] assessed performance of different machine learning algorithms in simulated environment called Robocode for single and multi-agent case. Neural network-based proposed architecture had shown superior results in simulated environment than others. Winning percentage verses rounds performance has been evaluated and proves that calculation of reward functions in each node in environments can be solve more accurately in neural network-based approaches. Kormushev et al. [13] have proposed time manipulation technique which improves learning in RL by minimizing fault tolerance and provides state exploration chance in better way than conventional RL algorithms. Results achieved in simulated environments showed that time manipulation algorithm has achieved better results in terms of best trail steps, benchmark trial steps and unique state visited. This work can be further explored to design other RL algorithms factors which can help to design better policy and reward functions.

Shibuya et al. [14] have discussed complex valued neural network which improves reinforcement learning. This experimental study investigates the use of eligibility traces in complex valued reinforcement learning. The authors provide insights into the benefits and limitations of this technique, contributing to the advancement of RL algorithms.

Lizotte et al. in this work [15] have done study on lineal programming (LP) especially in case of dynamic programming (DP). In LP, importance of value function does not affect too much except it is useful in primary problem cases. Authors presents a novel approach to solving Markov decision processes (MDP) using dual representations. The

authors introduce a new type of linear program that exploits the structure of the problem to improve computational efficiency. Their approach is shown to outperform traditional methods in both synthetic and real-world applications. Overall, the paper offers valuable insights into the potential of dual representations for dynamic programming, and its findings have important implications for the development of more efficient algorithms in reinforcement learning.

Authors [16] have proposed self-organizing decision tree which works on split estimation and tree growing approaches using tree reinforcement learning (TRL). This method reduces greedy approach on decision tree calculation by long term inducer estimator. Experiments show that error rate and tree size has been reduced in proposed method case than CART (classification and regression tree) using five datasets mentioned in paper. This work can be good choice to minimize exploitation and exploration in search of goal. Article [17] focuses on routing packets efficiently in dynamic network. Reinforcement learning can be suitable approaches in dynamic environmental. The use of reinforcement learning (RL) in network routing has shown promise in adapting to dynamic network changes. RL algorithms (proposed method) can learn and optimize network routing policies based on feedback from the network environment. However, challenges such as scalability and ensuring stability and fairness in the network must be addressed to fully utilize the potential of RL in network routing.

Asgharnia et al. [18] have proposed multi-objective fuzzy Q learning (MOFQL) to solve such real-world problems which contains different goals. This approach has been applied in gaming to achieve goals. Value function at each state-action pair is computed with fuzzy inference system (FIS). Temporal difference (TD) did fuzzy rules update mechanism. This paper draw attention to work in multi-agent and control system applications. This article [19] is a review work done by their authors of multi-agent RL. Authors have identified few problems in this case and then they figure out their solutions in context of stochastic gaming world. Bellman' heritage from simple agent learning to multi-agent learning has been discussed and explored critical review on this. The study [20] explores the use of reinforcement learning mechanisms and common knowledge fields in heterogeneous agent systems. It proposes a novel approach to enable agents to learn from their experiences and interactions with the environment and other agents. The study provides insights into the potential of this approach for improving the performance and adaptability of heterogeneous agent systems using sensory and behavioral input. Simulated results shows that learning in common knowledge field has shown positive aspect of it.

Kaelbling et al. in their work [21] have done survey on RL related with central concept, hidden Markov model (HMM) as foundation, discussing tradeoff between exploration and exploitation, speeding the learning etc. This paper could be a basic one for beginners to understand mathematical reinforcement learning models. Low et al. [22] have implemented improved Q-learning path finding algorithm for mobile robot. Effectiveness of algorithm has been tested with variable number of obstacles and their performance has been compared with traditional Q learning. Proposed algorithm has shown better time complexity than Q-learning. This work can be further improved with computational aspect and by creating more complex environments for multi-agent and muti-objective scenarios.

Path finding in optimal way [23] remains complex topic in the case of different kind of agents like humanoid, mobile robot, and any autonomous systems. Authors have proposed effective Q-learning (EQL) which calculates reward function closely and exploration and exploitation of optimum path dynamically. Computation results and analysis show that computation time and path length has been minimized in EQL for twelve different environments. This paper [24] proposes a method for path planning in 3D workspaces for robot arms using Q-learning and neural networks. The proposed method involves using computer vision to capture the current state of the environment and then using Q-learning to select the next best action. The neural network is used to predict the expected reward for each possible action, improving the efficiency of the learning process. Overall, the paper provides an interesting approach to solve the problem of path planning in 3D workspaces for robot arms.

Wang et al. in this study [25] investigated simulation-based RL on distributed very large-scale integrated scale (VLSI) to find optimum size of bounded time. Dynamic programming connection with RL has been used to find optimum policy. This work gives options to use RL in automatic chip designing and circuit management. Frank et al. [26] have proposed RL framework by creating more complex learning environment called iCub humanoid robot. This was called curious agent which was planning motion for humanoid in defined environment. Paper solves artificial curiosity by checking how much fast agent is learning. Paper is good enough to learn about path planning and trajectory generation using RL.

Wen et al. [27] have proposed a novel algorithm called fuzzy Q-learning (FQL) obstacle avoidance techniques for humanoid in unknown environment. FQL, Q-learning (QL) and optimized Q- learning has been compared with Checkboard model. Results of simulation shows that FQL is better than other traditional RL approaches. Bae, H. et al. in this article [28] proposed hybrid approach (deep Q learning plus CNN) to path find problem in multi-agent environment. This work minimizes communication effect among robots using convolutional neural network (CNN) with image identification. Simulated environment containing C++ and Linux has shown that invented approach was far better than traditional

Q-learning. Article [29] investigates shaping trend in RL in terms of reward function, local optimization, global goal achievement, policy upgradation, preparation for unknown environment, state-action value and many others influential parameters which accelerates learning in this domain. Authors have suggested to improve different aspects in learning fundamentals.

Authors [30] have focused on solving two major problems of RL, state-action estimation and selecting best move in large space of size 2^{40}. Value function approximation has shown promising results in simulated environments. Mahadevaswamy et al. [31] have drawn attention on robot 3D automatics mapping inside home or buildings. Autonomous navigation and 3D mapping in path finding and trajectory generation play significant role to learn from mistakes, trial- error RL basics. Authors [32] have compared query base learning agent (QA) and temporal difference network (TDN) based on learning rate, discount rate and memory usage. Result shows that QA has achieved better result than TDN in controlled environments. This work can be further explored using different deep reinforcement learning algorithms with different constraints in multi-agent, dynamic environment.

Morimoto [33] et al. has proposed bipedal walking for 3 link and 5 link robot simulators. Learning is based on Poincare model and selection of actions from computed value functions. Results showed biped walking before, after and in middle on using 3-trials of learning. The paper presents some promising experimental results on a simulated humanoid robot, showing that the proposed approach can learn to walk with reasonable performance. However, there are some limitations to the paper that should be considered. First, the paper focuses exclusively on simulations and does not provide any experimental results on a physical robot. While simulations can be a useful tool for testing algorithms, they are not always representative of real-world performance. Therefore, it is important to evaluate the proposed approach on a physical robot to assess its practicality and robustness. Raj and Kumar [34] has applied reinforcement learning to control bipedal walk using inverted pendulum concept. Double inverted pendulum was tested in simulated environments for different learning rate, discount rate and action (clock state wise and anti-clockwise). Limit cycle has been achieved by achieving pole angle and pole angular velocity zero. The paper is a valuable contribution to the field of robotics and can be of interest to researchers and practitioners working on bipedal walking control.

Peters et al. [35] have discussed the reinforcement learning for humanoid robotics where generally they have classified learning into three major methods (i) greedy method (ii) vanilla policy gradient method and (iii) natural gradient method. Authors have proposed natural actor-critic algorithm which achieves local minima for cost function. This algorithm has achieved better results from others in nonlinear systems of humanoids. Paper is good mathematical validation to understand all discussed methods. Number of episodes verses expected reward(J(theta)) has been evaluated. Zhang et al. in this [36] work have proposed a learning framework called LORM (learn and outperform the reference motion) considering different environments (plains, slope, uneven terrains, and push factor) for biped's gait control. Framework has been crafted in all possible way to optimize humanoid velocity than existing methods on Darwin-op robot simulator. Two major task was performed and validated (walking as fast as, tracking specific velocity). Results shows that tracking velocity rate was more than 95% while maximum speed with 0.488 m/s was achieved. This paper is clearly written and good enough to do further work on this.

Canese et al. [37] have described different challenges in multi-agent environments. Recent research has already shifted towards distributed environment which mimics real environments of different agent communication to each other to optimize learning policy. Multi-agent reinforcement learning (MARL) is the future of agent learning. Developing algorithms in this will be highly applicable in different field of autonomous and learning sectors. This paper [38] does systematic review of different reinforcement learning which are model free and model based. Different current learning for different applications has been explored and future scope of meta learning, automated machine learning (AML) and self-learning discussion has been discussed.

We have done more detailed analysis of related work used in this paper which covers different algorithms used, type of learning, functions used and their applications in different sectors (Table 1). This will further help us to define few issues and new emerging trends of learning for future used and have used by research communities.

Research Gap

Reinforcement learning (RL) is a popular technique used in robotics and control systems to generate trajectories for agents operating in dynamic environments. The use of Q-learning in RL has been widely explored in trajectory generation for various applications, including robotics, control systems, and game AI. However, there are still some research gaps in this area that require further investigation.

One of the major research gaps in trajectory generation using Q-learning is the scalability of the technique. Q-learning is known to be computationally expensive, and it may not be suitable for real-time applications that require fast trajectory generation. Therefore, researchers need to investigate ways to optimize the Q-learning algorithm to make it more scalable and efficient for real-time applications [3].

Table 1 Comparative study of review work [2–37]

Approaches/algorithm	Agent	Reward function	Outcome	Application	References
Swarn RL	Multi-agent and single agent	Optimal policy	Proposed algo is better in both cases where in one goal is fixed and another one in which goal is changing in 10×10 grid environments using episodes and no. of actions	Humanoid, mobile robotics, shortest path finding	[2]
DNN (dynamic neural network)	Multi-agent	Value based	Application of multilayer feedforward (MLFL) neural network into DNN (dynamic neural network) has been described	Intelligent robot, neuroscience	[3]
RL_Spider (vertical search engine)	Single agent (spider)	Value and policy based	RL_Spider and Nuts searching comparison on noted websites showed that spider-based learning was better on state, action, reward	Path finding. robotics	[4]
Simulation based dynamic programming	Multi-agent/single agent	Value based	Used to find optimum size of bounded time optimally	Control systems	[5]
Rodent and Hippocampal inspired learning	Multi-agent	Value based	Place and grid cells help robot to navigate unexplored path using VTE (vicarious trial and error) model	Path optimization, neuroscience. Biohybrid systems	[6]
Hybrid learning architecture	Single/multi-agent	Value based	Learning framework demonstrates that integration of RL and imitation both can be useful. More experiments should be performed	Bipeds, Bioscience	[7]
Novel framework from partially-observable Markov decision process (POMDP)	Multi-agent	Value based	Proposed learning farmwork helps to generate internal reward and finally help to achieve global reward for organization (agent)	Gaming, education, and finance sector	[8]
Trajectory feedback learning system	Single/multi-agent	Value based	Stronger feedback learning system based on least square error (LSE) and Thompson sampling (unknown transition model) has help to achieve the learning goals better	Control systems UAV, multi-stage UX interface	[9]
RL-DNSA (dynamic network self-optimization algorithm)	Multi-agent	Value based	Universal mobile telecommunication system (UMTS) and WLAN as agent have shown better learning in terms of Q learning, bandwidths, cell capacity and coverage etc	Telecommunication sector, operation management	[10]
ANN-RL	Multi-agent	Value based	Robocode simulator was used to check the results which shows that ANN (artificial neural network) based RL is better than other approaches. Results need to verify on other simulators for deep RL	Gaming, robotics	[12]
Time manipulation based learning	Multi-agent	Value based	Going forward and backward in simulated environment leads to new learning which is applied in real scenario. Paper shows how to avoid failure searching and explore space exploration	Gaming, failure avoidance problems	[13]
Complex valued NN	Multi-agent	Value based	Eligibility tracing in unknown environment helps to achieve global results in certain constraints	Autonomous systems, self-driving car, search optimization	[14]

Table 1 (continued)

Approaches/algorithm	Agent	Reward function	Outcome	Application	References
Dual approach to DP (dynamic programing) and RL	Multi-agent	Policy and value based	DP and RL both helps to achieve better learning with probabilistic and better exploitation	Sequential decision making, linear programming	[15]
Self-organizing tree learning	Single/multi-agent	Value based	Tree induction-based learning to get better search space and reward point at each node has been computed, proposed algo was better than others decision trees	Search space optimization, high performance computing	[16]
Routing RL	Multi-agent	Value based	Packet routing in dynamic traffic condition has been evaluated in simulated environment and RL is better than other traditional learnings	Route optimization, mathematical optimization	[17]
Fuzzy multi-objective RL (MOFQL)	Multi-agent	Value plus policy based	Proposed learning algorithm has shown positive results to achieve multi objective in given problems	Gaming, multi-objective recommendation system, economics, logistics	[18]
Heterogeneous RL	Heterogeneous multi-agent	Value based	Agents learning based on proposed learning rules has shown extended results in simulated environments	Mobile robotics	[20]
Improved Q-learning	Single agent (three wheeled mobile robot)	Value based	Proposed approach has uses FPA (flower pollination algo) to initialize q table and method learning was tested with different simulated test cases. Computation time was reduced	Mobile robotics path optimization	[22]
Efficient Q-learning (EQL)	Mobile robot as multi-agent	Policy plus value based	EQL performance was better than other Q-learning in terms of path length, computation time and safety parameters and state of arts	Robotics, navigation systems	[23]
Computer vision neural network RL	Single/multi-agent	Value based	Computer vision for object detection, Q learning for goal finding and neural network for training has shown collectively impressive results	Humanoids, gait analysis, AR-VR, deep RL, 3D construction	[24]
Adaptive time wrap RL	Single agent	Value based	Proposed algorithm shows optimum solution for time wrap in vlsi simulator	Circuit designing. Optimization problems	[25]
RL framework	iCub humanoid as single agent	Value based	Authors tries to simulate learning for humanoid unlike from toy agents. Intelligence and curiosity to learn was monitored	Humanoid robot, motion planning	[26]
Fuzzy Q learning (FQL)	Single agent	Value based	Obstacle avoidance problem was optimized and found that FQL was better than other RL	Humanoid, obstacle avoidance, path planning	[27]
Deep q learning with CNN	Multi agent	Value based	Performance parameters like epochs, goals states, episodes comparison show better results than tradition q learning	Path planning, robotics	[28]
Shaping RL	Single/multi-agent	Value based	Research gap between continuous and discrete methods has been applied in RL	Discrete and continuous problems	[29]

Table 1 (continued)

Approaches/algorithm	Agent	Reward function	Outcome	Application	References
Novel approximation method	Multi-agent	Value based	Proposed algo helps to calculate value (reward) function with the help of product of expert method	Optimization problems, model learning, gaming	[30]
Robotic mapping	Single/multi-agent	Value /policy based	With the help of navigation and mapping techniques, 3D map system was built and learning was fruitful to understand 3D mapping for indoor environments	Autonomous systems, agent learning, smart city, and home	[31]
Query base self-learning (QA)	Single/multi-agent	Value based	QA was better than TDN in terms of discount rate, memory usages and learning rate	Control problems, robotics, weather forecasting	[32]
Model based RL	Multi-agent	Value based	Different walking speed before, after, and in middle in fix and dynamic rails has been checked in simulated environment to test learning of robot. Three and five link simulators have been used to show case results	Bipeds, gait analysis	[33]
Double pendulum-based Q learning	Single agent as humanoid	Policy based	Human walking inspired learning is being investigated using RL. Double inverted pendulum was key approach	Gait analysis, humanoid robotics	[34]
Reinforcement learning	Single/multi-agent	Value based	Different traditional reinforcement learning has been compared and their applicability in humanoids has been discussed	Humanoid robotics	[35]
Learn and outperform the reference motion (LORM) framework	Single agent as humanoid robot	Value based	With the help of proposed LORM framework, humanoid robot walking is controlled in Darwin -op plate from. Learning was done successfully	Humanoid, human locomotion, gait control	[36]
Reinforcement learning	Single and multi-agent	Value and policy based	In this paper all type of RL were analyzed. Application and related sectors were discussed	Agent learning, control systems, robotics	[37]

Reference no. [1, 11, 13, 21, 38] (review paper) were not included as they do not contribute novel work

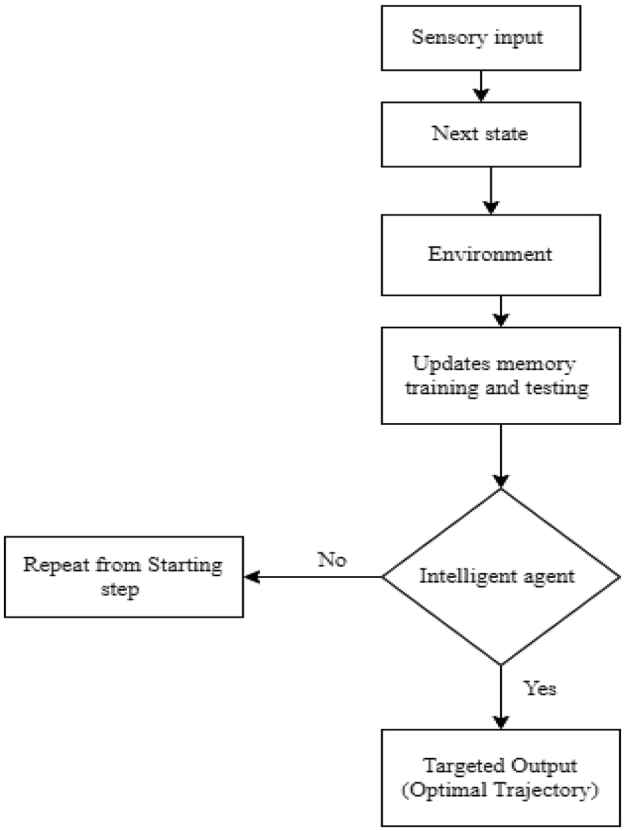

Fig. 2 Block diagram of learning agent for optimal trajectory

We have worked on this research gap to propose optimal Q learning which is better than Q-learning.

Another research gap in this area is the exploration of different reward functions [30]. The reward function is a critical component of Q-learning, and it plays a significant role in determining the trajectory generated by the agent. Researchers need to explore different reward functions and investigate their effects on the trajectory generation process. This will enable them to identify the best reward function that can optimize the performance of the agent. Furthermore, the exploration of deep reinforcement learning (DRL) in trajectory generation [12] is another research gap that needs to be addressed. DRL is a new area of RL that combines deep neural networks with RL algorithms to enable agents to learn complex tasks. Researchers need to explore the use of DRL in trajectory generation and compare its performance with traditional RL techniques like Q-learning.

Lastly, the evaluation of trajectory generation algorithms is an important research gap that needs to be addressed. Researchers need to develop evaluation metrics that can be used to assess the performance of trajectory generation algorithms. This will enable them to compare different algorithms and identify the best algorithm for specific applications [30, 32].

Overall, addressing these research gaps will help advance the field of trajectory generation using reinforcement Q-learning and enable the development of more efficient and effective trajectory generation algorithms for various applications. Research community must introduce more realistic and optimal algorithm, frameworks and design reward or penalty functions to handle real-time scenarios.

Research Challenges in Trajectory Generation [37]

Trajectory generation is an important task in reinforcement learning (RL), where an agent learns to perform a task by interacting with an environment and receiving feedback in the form of rewards. The goal of trajectory generation is to generate a sequence of actions that lead to a high cumulative reward. Here are some research challenges in trajectory generation in RL:

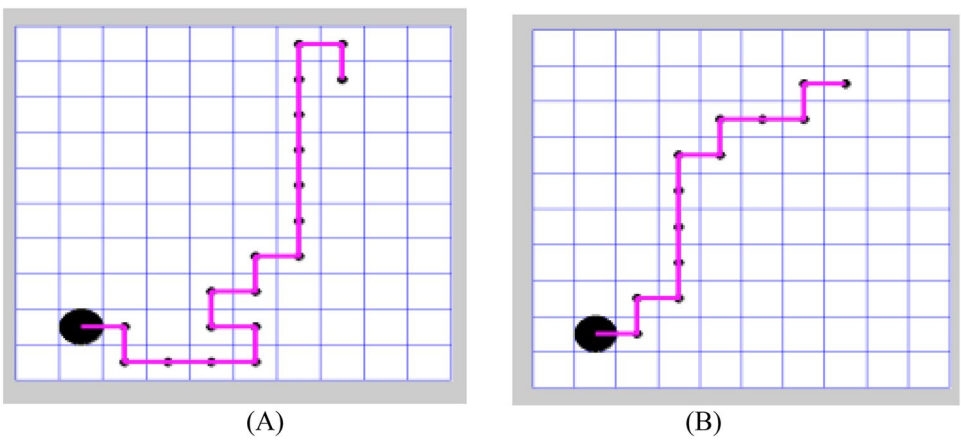

Fig. 3 In the grid world of 10×10, starting from (2, 8), an Agent moves aiming the goal at (9, 2). Left: Single path from 50 trial. Right: B route of the optimal trajectory to the goal

Exploration–Exploitation Tradeoff (E–E Effect): Trajectory generation algorithms need to balance E–E effect. Exploration is necessary to discover new and potentially better actions, while exploitation is necessary to leverage the information already learned to maximize reward. A challenge is to find a balance between exploration and exploitation that leads to optimal performance.

Scalability: Many RL tasks involve high-dimensional state and action spaces, which can make trajectory generation computationally expensive. Scalability is a significant challenge in RL trajectory generation, and researchers are exploring ways to optimize the computational efficiency of trajectory generation algorithms.

Transferability: Transfer learning is an essential aspect of RL, where an agent trained in one environment can leverage its knowledge to perform well in a different but related environment. Trajectory generation algorithms need to be transferable across different environments, which requires generalization and adaptation to new environments.

Handling uncertainty: RL is inherently uncertain, and trajectory generation algorithms need to handle this uncertainty to generate robust and reliable trajectories. This challenge includes modeling and predicting uncertainties in the environment, such as sensor noise, modeling errors, and stochasticity in the environment.

Safety and robustness: Trajectory generation algorithms need to ensure the safety and robustness of the learned policies. Safety is a critical concern when deploying RL systems in real-world applications, and trajectory generation algorithms need to consider safety constraints while optimizing for reward.

Addressing these challenges requires a combination of theoretical and practical approaches, including designing new algorithms, developing more efficient computational methods, and collecting more diverse and realistic datasets.

Proposed Method and Algorithm

The generalization of the environment and state action (s, a), which control sensory input during training sessions, are discussed in this section. We must model different nodes as states across a specific environment to achieve this goal. It carried out a random state action and began looking for options within the four options provided (left, right, up, and down) to archive a new state. Repeat starting at step one if the current condition is comparable to the prior state action. After verifying that the goal has been reached, move on to the next stage; if not, update the look-up table.

To train the various nodes of the network in a given environment with the best possible policy rewards and penalties, we used Q-learns, which has the model-free enhancement study characteristic. Through numerous trial-and-error interactions, this agent learns how to behave toward the Marko process optimally. Every iterative process is stored in the provided look-up table, and during testing, a random value is chosen from the available actions. All nodes in a particular environment that uses the Q-learning algorithm train using objective function approximation (Q function) similar to a satellite state-action input pair. Sensory input and target output for the specified challenge is involved in the physical activity during network training. Based on state-action pair values, the Q-learning algorithm is valued as a reward at the start of the training process. Classifications of occurrences and repeated knowledge processes occur for the nth time in Q learning.

Base Q-Learning [23]: Umbrella Term for Reinforcement Learning

Q-learning is a type of reinforcement learning algorithm that learns to make decisions in an environment by trying different actions and observing the rewards that result from those actions. Q-learning uses a table called a Q table to store the expected rewards for each state-action pair, and uses a function called the Q-function to determine which action to take in a given state.

Steps:

1. Initialize the Q-table with all zeros for each state-action pair.
2. Set the learning rate, gamma, and epsilon values.
3. Repeat for each episode:
   - 3.1 Initialize the environment and set the initial state.
   - 3.2 Repeat for each step in the episode:
     - i. Choose an action based on the current state and the epsilon-greedy policy.
     - ii. Take the action and observe the resulting reward and next state.
     - iii. Update the Q-table value for the current state-action pair using the Q-learning formula

       Q (s, a) = Q (s, a) + alpha * (reward + gamma * max (Q (next_state, a)) - Q (s, a))// any time t
     - iv. Set the current state to the next state.
   - 3.3 Decrease epsilon and alpha values as the number of episodes increases.
4. Use the Q-table to determine the optimal policy for the given environment.

In the Q-learning formula, alpha is the learning rate, which determines the weight given to new information versus old information, gamma is the discount factor, which determines the importance of future rewards, and max (Q (next_state, a)) is the maximum expected reward for the next state.

Optimal Trajectory Generation algorithm:

Steps:

1. In a given grid search space, randomly generates the initial state (s_n) as input. // $s_n = \{s_1, s_2, s_3, \ldots\}$
2. Check for available actions (a_n) in search space (Grid)
3. Select any random action
4. If action ($a_{1)}$ leads to same as previous state (s_{n-1})
5. then start step one
6. when the target is achieved
7. Store iterative values (s_n, a_n) in a transient array
8. Updates transient array $r_1 = r\gamma^{j-k}$
9. Creates (s_{n+1}) state using the state-action, next episode repeats step 1 to step 9
10. Until the objective is accomplished

Pseudo code for proposed algorithms:

```
Initialize the Q-table with zeros for all state-action pairs
Repeat the following for each episode:
    Initialize the state s
    Repeat the following until the episode ends:
        Choose an action a from state s using an exploration policy
        Take the action a and observe the reward r and the next state s'
        Update the Q-value for the state-action pair (s, a) in transient array r1 using the Q-learning update rule:
            Q (st, at) = Q (st, at) + alpha * (r + gamma * max (Q (s't, a't)) - Q (st, at))// sn+1
            Assign s to s' as new state
    End Repeat
End Repeat
```

Learning Agent for Optimal Trajectory

The proposed framework will fulfill the requirement of a learning mechanism.

Working Methodology

At very first step, we plan to describe (Fig. 2) surroundings and the state input. We refer to the state having a specific value s. Thus, it is introduced to the state-action pair throughout training. Currently, we want to develop a training module. To achieve a new state, we first start with a random beginning state and look for actions that can be taken as environmental input. If the state-action that was previously taken is equal to the current state, repeat the process to reach the starting state. Now check the target status; if it was accomplished, update the following episode. If not, save the state-action pair (s_t a_t). Using the state-action pair, create the following state (s_{t+1}).

Define the agent as a pair of states and actions. Then choose the train, which opens a new window with an array editor network. You can visualize the learned agent at this point as a Q table. It can also be visualized as a chosen route or direction. Now we can specify the aim and inputs using state-action pair (100 × 4) for 10 × 10 grid world. We chose Q-learning because it essentially searches for the best policy without any prior knowledge of the system, which is a feature of model-free RL. Gaining experience with action consequences function learning gives agents the ability to learn to act optimally in Marko domains (function learning typically saves Q value relative to every state activity in a look-up table).

The state-action value is gradually updated by the iterative training algorithm (Q-learning). The agent can be trained as a state-action pair to estimate a function (Q function). The state input and target output are essential for the training procedure. To estimate the state-action pair values using a discount rate-learning methods use reward function during training to evaluate state-action value in terms of reward value.

Results Analysis and Comparisons

In this paper, various common trajectory planning tasks such as obtaining a moving target and avoiding obstacles by taking viable actions in the grid world problem has been implemented. Learning accuracy can be evaluated using the performance of the optimal trajectory algorithm. ET measures the effectiveness of goal monitoring in the specific goal position as well as the overall percentage of correctly evaluated training rate [32].

$$\text{Trajectory efficiency (Et)} = \left[1 - \frac{\text{Minimum step}(j) - \text{Total count step}(k)}{\text{Total path}}\right] \times 100,$$

where the minimum count step is j and k are the total count step.

Learn Optimal Trajectory from the Episode

We have implemented trajectory generation of agent in MATLAB where agent can move one step upward, downward, to the right, or to the left. Impeded by the border of the grid world, one option is to refrain from taking any action and instead rely on chance to determine the subsequent course of action. The agent would keep doing this until the target is attained. The maximum number of steps should be determined in relation to the grid world's size. We will now start by looking at the consequences of an agent's random action in Fig. 3, indicated below.

The accuracy of the trajectory efficiency with respect to several episodes is shown below in Table 2. These outcomes have been compared and presented in Fig. 4.

The learning accuracy of the trajectory algorithm is over 96.70%, as shown in Table 2, and most of them are greater than the Q-learning algorithm. As a result of all the experimental procedures, we observed that the trajectory algorithm.improves the q-learning method. The optimal trajectory algorithm and the q-learning algorithm have been compared in this case, and it has been determined that the trajectory algorithm is superior in terms of target tracking, learning time, and memory use. Finally, we compare and present the graph of comparison as shown below.

From above (Fig. 5) it is clear that proposed algorithm is having better trajectory efficiency than Q-learning.

Based on the table of trajectory efficiency results (Tables 2, 3, Figs. 6, 7), we can make the following observations:

(i) Overall, the average trajectory efficiency (Fig. 6) tends to increase as the number of episodes increases. The average trajectory efficiency is 75.98% for episode 1000, 80.40% for episode 3000, 77.68% for episode 5000, 90.54% for episode 7000, 91.76% for episode 9000, and 93.56% for episode 11,000.
(ii) The performance of the agent varies across trajectories (Fig. 7). For example, trajectory T5 has the highest efficiency for episodes 1000, 3000, 7000, and 9000, while trajectory T1 has the highest efficiency for episodes 7000 and 11,000.
(iii) Trajectory T5 generally performs well across episodes, with above average efficiency for all but one

episode. This suggests that the agent is able to consistently navigate this trajectory effectively.

(iv) Trajectory T1 has lower efficiency for other episodes except 7000- and 11,000-episode number. This suggests that the agent may be struggling with certain aspects of this trajectory, or that the rewards in this trajectory are more difficult to obtain.

(v) Trajectory T3 has relatively low efficiency across all episodes. This suggests that the agent may be having difficulty navigating this trajectory or obtaining the rewards it needs to perform well.

(vi) Proposed work has shown improvement than existing work. Through accuracy is not good.

Table 2 Comparison table between different episodes for optimal trajectory algorithm

Trajectory efficiency	For episode $E=1000$	For episode $E=3000$	For episode $E=5000$	For episode $E=7000$	For episode $E=9000$	For episode $E=11{,}000$
T1	59	94.80	79	94.83	79.20	95.71
T2	69.58	66	60	82	93.32	94.82
T3	73.30	64.85	95.76	94.63	72.41	83.60
T4	85	73.65	96.63	82.81	93.91	96.70
T5	93.80	94.95	60.10	86.91	93.23	96.50
Average	75.98%	80.40%	77.68%	90.54%	91.76%	93.56%

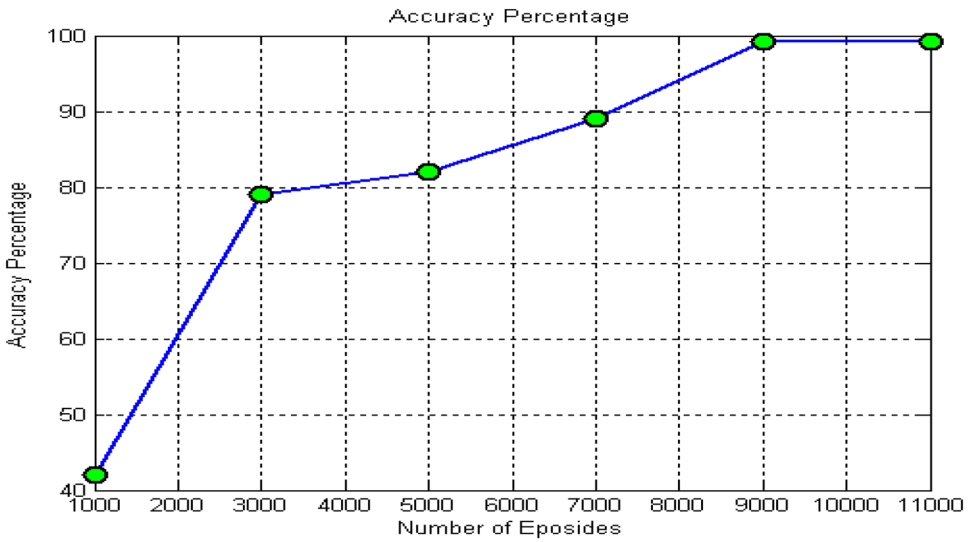

Fig. 4 Comparison view of different episodes

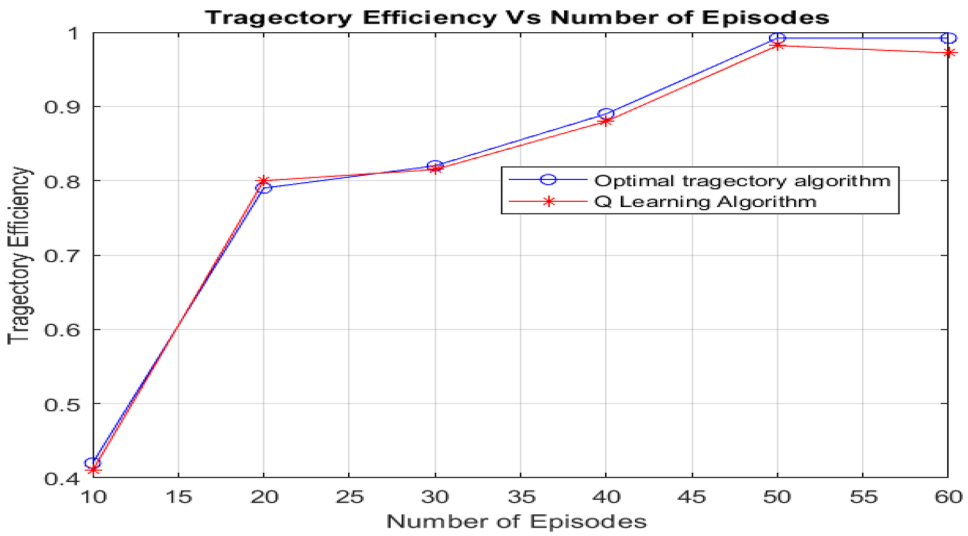

Fig. 5 Performance Comparison graph between optimal trajectory and q-learning algorithm

Table 3 Comparative study of proposed and existing work

Method	Training accuracy %	Reference
QA algorithms	96.20	[32]
Proposed algorithm	96.70	

Overall, these results suggest that the agent's performance is improving over time, and that its ability to navigate different trajectories varies. Further analysis would be needed to determine the causes of these variations and to identify ways to improve the agent's performance.

Conclusion

In this, research paper focuses on novel algorithm for optimal trajectory generation using Q learning based on grid world problem. We show that without prior knowledge of the system agent can make decision over given environment. Using this method this algorithm is suitable for better decision making along with optimal trajectory with 96.70%. Proposed algorithm to learn shortest trajectory path under each given episodes, repetitive loops in episodes are removed to speed up convergence. Based on optimal trajectory algorithm, we got better results and accuracy in term of trajectory efficiency over other existing Q-learning algorithm.

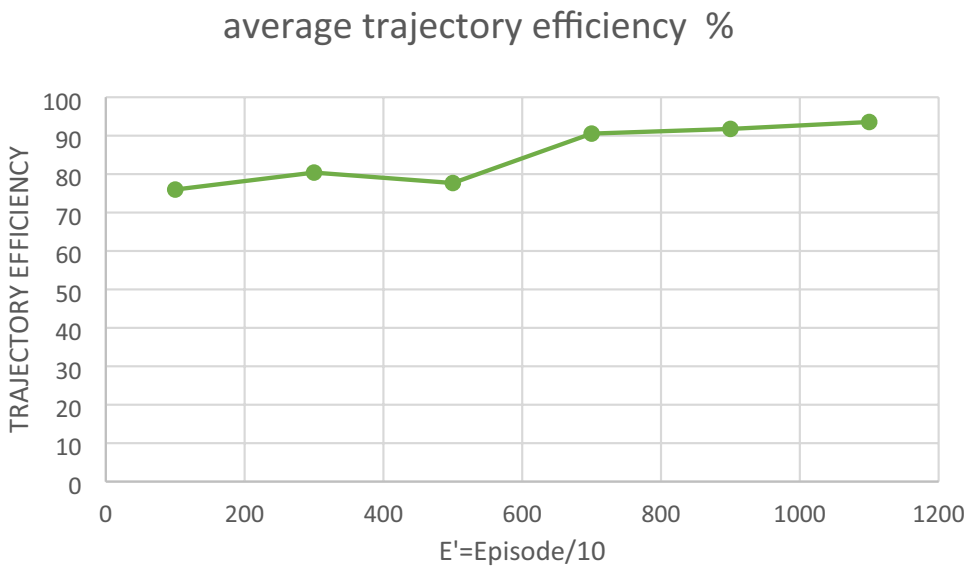

Fig. 6 Average trajectory efficiency verses episode

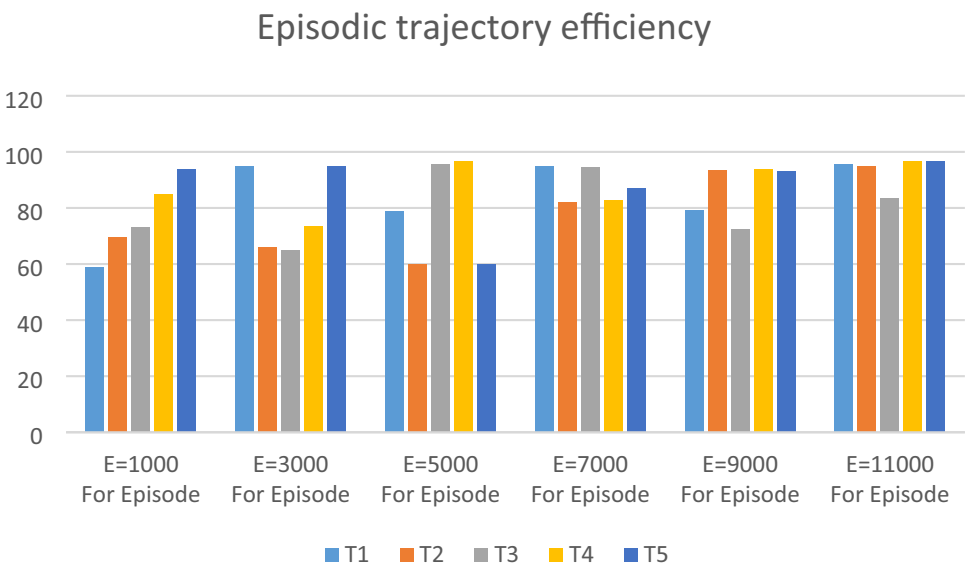

Fig. 7 Learning accuracy of proposed trajectory algorithm

In the future work, we can make optimal trajectory algorithm more effective using combining multiple learners and other decision techniques as computational sequence learning. This work can be further extended using different alternative of Q- table like neural network, function approximation, hash table and binary tree. Finding optimal trajectory for different agent in different environment based on different sector will remain challenging in future.

Author Contributions VBS Assistant Professor, MANIT, Bhopal, Madhya Pradesh, India. DKM Assistant Professor ASET Amity University, Gwalior, Madhya Pradesh, India.

Funding This study is not funded.

Declarations

Conflict of interest On behalf of all authors, corresponding author states that there is no conflict of interest.

References

1. Ding Z, Huang Y, Yuan H, Dong H. Introduction to reinforcement learning. In: Deep reinforcement learning: fundamentals, research and applications. 2020. p. 47–123.
2. Iima H, Kuroe Y. Swarm reinforcement learning algorithms based on Sarsa method. In: 2008 SICE annual conference. IEEE; 2008. p.2045–2049.
3. Yadav AK, Sachan AK. Research and application of dynamic neural network based on reinforcement learning. In: Proceedings of the international conference on information systems design and intelligent applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012. Berlin: Springer; 2012. p. 931–942.
4. Quan L, Zhi-ming C, Yu-chen F. The research on the spider of the domain-specific search engines based on the reinforcement learning. In: 2009 WRI Global congress on intelligent systems, vol 2. IEEE. 2009. p. 588–592.
5. Wang J, Tropper C. Optimizing time warp simulation with reinforcement learning techniques. In: 2007 winter simulation conference. IEEE. 2007. p. 577–584.
6. Santos-Pata D, Zucca R, Verschure PF. Navigate the unknown: implications of grid-cells “mental travel” in vicarious trial and error. In: Proceedings 5 Biomimetic and biohybrid systems: 5th international conference, living machines 2016, Edinburgh, UK, July 19–22, 2016. Springer International Publishing; 2016. p. 251–262.
7. Hamahata K, Taniguchi T, Sakakibara K, Nishikawa I, Tabuchi K, Sawaragi T. Effective integration of imitation learning and reinforcement learning by generating internal reward. In: 2008 Eighth international conference on intelligent systems design and applications, vol 3. IEEE; 2008. p. 121–126.
8. Taniguchi T, Tabuchi K, Sawaragi T. Role differentiation process by division of reward function in multi-agent reinforcement learning. In: 2008 SICE annual conference. IEEE. 2008. p. 387–393.
9. Efroni Y, Merlis N, Mannor S. Reinforcement learning with trajectory feedback. In: Proceedings of the AAAI conference on artificial intelligence, vol 35, no 8. 2021. p. 7288–7295.
10. Feng Z, Tan L, Li W, Gulliver TA. Reinforcement learning based dynamic network self-optimization for heterogeneous networks. In: 2009 IEEE Pacific rim conference on communications, computers and signal processing. IEEE. 2009. p. 319–324.
11. Sivamayil K, Rajasekar E, Aljafari B, Nikolovski S, Vairavasundaram S, Vairavasundaram I. A systematic study on reinforcement learning based applications. Energies. 2023;16(3):1512.
12. Ulusoy Ü, Güzel MS, Bostanci E. A Q-learning-based approach for simple and multi-agent systems. In: Multi agent systems-strategies and applications. IntechOpen. 2020.
13. Kormushev P, Nomoto K, Dong F, Hirota K. Time manipulation technique for speeding up reinforcement learning in simulations. 2009. arXiv:0903.4930.
14. Shibuya T, Shimada S, Hamagami T. Experimental study of the eligibility traces in complex valued reinforcement learning. In 2007 IEEE international conference on systems, man and cybernetics. IEEE. 2007. p. 1630–1635.
15. Lizotte D, Wang T, Bowling M, Schuurmans D. Dual representations for dynamic programming. 2008.
16. Hwang KS, Yang TW, Lin CJ. Self organizing decision tree based on reinforcement learning and its application on state space partition. In: 2006 IEEE international conference on systems, man and cybernetics, vol 6. IEEE. 2006. p. 5088–5093.
17. Khodayari S, Yazdanpanah MJ. Network routing based on reinforcement learning in dynamically changing networks. In 17th IEEE international conference on tools with artificial intelligence (ICTAI'05). IEEE. 2005. p. 5.
18. Asgharnia A, Schwartz H, Atia M. Multi-objective fuzzy Q-learning to solve continuous state-action problems. Neurocomputing. 2023;516:115–32.
19. Shoham Y, Powers R, Grenager T. Multi-agent reinforcement learning: a critical survey, vol 288. Technical report, Stanford University. 2003.
20. Kawakami T, Kinoshita M, Kakazu Y. A study on reinforcement learning mechanisms with common knowledge field for heterogeneous agent systems. In: IEEE SMC'99 conference proceedings. 1999 IEEE international conference on systems, man, and cybernetics (Cat. No. 99CH37028), vol 5. IEEE. 1999. p. 469–474.
21. Kaelbling LP, Littman ML, Moore AW. Reinforcement learning: a survey. J Artif Intell Res. 1996;4:237–85.
22. Low ES, Ong P, Cheah KC. Solving the optimal path planning of a mobile robot using improved Q-learning. Robot Auton Syst. 2019;115:143–61.
23. Maoudj A, Hentout A. Optimal path planning approach based on Q-learning algorithm for mobile robots. Appl Soft Comput. 2020;97: 106796.
24. Abdi A, Ranjbar MH, Park JH. Computer vision-based path planning for robot arms in three-dimensional workspaces using Q-learning and neural networks. Sensors. 2022;22(5):1697.
25. Wang J, Tropper C. Optimizing time warp simulation with reinforcement learning techniques. In 2007 winter simulation conference. IEEE. 2007. p. 577–584.
26. Frank M, Leitner J, Stollenga M, Förster A, Schmidhuber J. Curiosity driven reinforcement learning for motion planning on humanoids. Front Neurorobot. 2014;7:25.
27. Wen S, Chen J, Li Z, Rad AB, Othman KM. Fuzzy Q-learning obstacle avoidance algorithm of humanoid robot in unknown environment. In: 2018 37th Chinese control conference (CCC). IEEE. 2018. p. 5186–5190.
28. Bae H, Kim G, Kim J, Qian D, Lee S. Multi-robot path planning method using reinforcement learning. Appl Sci. 2019;9(15):3057.
29. Erez T, Smart WD. What does shaping mean for computational reinforcement learning? In: 2008 7th IEEE international conference on development and learning. IEEE. 2008. p. 215–219.
30. Sallans B, Hinton GE. Reinforcement learning with factored states and actions. J Mach Learn Res. 2004;5:1063–88.

31. Mahadevaswamy UB, Keshava V, Lamani AC, Abbur LP, Mahadeva S. Robotic mapping using autonomous vehicle. SN Comput Sci. 2020;1:1–12.
32. Yadav AK, Shrivastava SK. Evaluation of reinforcement learning techniques. In: Proceedings of the first international conference on intelligent interactive technologies and multimedia. 2010. p. 88–92.
33. Morimoto J, Cheng G, Atkeson CG, Zeglin G. A simple reinforcement learning algorithm for biped walking. In: Proceedings. ICRA'04 IEEE international conference on robotics and automation, vol 3. IEEE. 2004. p. 3030–3035.
34. Raj S, Kumar CS. Q learning based Reinforcement learning approach to bipedal walking control. In: Proc. iNaCoMM, Roorkee. 2013. p. 615–620.
35. Peters J, Vijayakumar S, Schaal S. Reinforcement learning for humanoid robotics. In: Proceedings of the third IEEE-RAS international conference on humanoid robots. 2003. p. 1–20.
36. Zhang W, Jiang Y, Farrukh FUD, Zhang C, Zhang D, Wang G. LORM: a novel reinforcement learning framework for biped gait control. PeerJ Comput Sci. 2022;8: e927.
37. Canese L, Cardarilli GC, Di Nunzio L, Fazzolari R, Giardino D, Re M, Spanò S. Multi-agent reinforcement learning: a review of challenges and applications. Appl Sci. 2021;11(11):4948.
38. Mehta D. State-of-the-art reinforcement learning algorithms. Int J Eng Res Technol. 2020;8:717–22.