



A modified Q-learning algorithm for robot path planning in a digital twin assembly system

Xiaowei Guo² · Gongzhuang Peng¹ · Yingying Meng¹

Received: 20 August 2021 / Accepted: 17 December 2021 / Published online: 10 January 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract

Product assembly is an important stage in complex product manufacturing. How to intelligently plan the assembly process based on dynamic product and environment information has become a pressing issue that needs to be addressed. For this reason, this research has constructed a digital twin assembly system, including virtual and real interactive feedback, data fusion analysis, and decision-making iterative optimization modules. In the virtual space, a modified Q-learning algorithm is proposed to solve the path planning problem in product assembly. The proposed algorithm speeds up the convergence speed by adding a dynamic reward function, optimizes the initial Q table by introducing knowledge and experience through the case-based reasoning algorithm, and prevents entry into the trapped area through the obstacle avoiding method. Finally, the six-joint robot UR10 is taken as an example to verify the performance of the algorithm in the three-dimensional pathfinding space. The experimental results show that the performance of the modified Q-learning algorithm is significantly better than the original Q-learning algorithm in both convergence efficiency and the optimization effect.

Keywords Path planning · Digital twin · Q-learning · Case-based reasoning

1 Introduction

Assembly is one of the most important parts of the industrial production process [1]. With the development of nowadays technology, humans are gradually replaced by industrial robots during the assembly process. Meanwhile, the development of industrial robots has also gone through several stages. Industrial robots using traditional assembly methods can only run under a set path. People usually use Cartesian coordinates to describe the robot's posture and obtain data through sensors, so as to obtain the relationship between the robot arms and the targets or obstacles. The D-H method has been used to extract the position parameters and joints' variables, combined with the rigid body pose description theory and

the inverse transformation method to solve the positive and negative kinematics [2]. This method requires plenty of calculation and a lot of human engagement. People have to plan the path manually for the robots, and what the robot can do is only repeatedly work on the route planned by the operator and feedback a series of displacement, velocity, and acceleration data to the operator. This method will cause the robot unable to adapt to the environment and to avoid obstacles autonomously.

However, with the current economic development, customers have stricter requirements for products. This is followed by more and more complex product designs and higher demand for assembly precision. The drawback of impotent real-time communication may lead to a negative influence on the verification of the component model as well as the evaluation of the overall product assembly, which will affect the whole product quality. The ability to autonomously avoid obstacles has also become more important for industrial robots. Thus, virtual assembly and machine learning were introduced. In virtual assembly, users can utilize various interactive devices to perform assembly operations on product components as in reality. During the operation process, the system provides real-time detection, management, and planning functions so that users can easily

✉ Gongzhuang Peng
gzpeng@ustb.edu.cn

Xiaowei Guo
xiaoweig@usc.edu

¹ National Engineering Research Center for Advanced Rolling and Intelligent Manufacturing, University of Science and Technology Beijing, Beijing 100083, China

² Daniel J. Epstein Dept. of Industrial & Systems Engineering, University of Southern California, Los Angeles, CA 90089, USA

analyze the feasibility of the assembly performance. After the process is completed, the system will take down all the information from the assembly process and generate reports and video recordings for continued analysis.

Since the most important parts in the virtual assembly system are collecting data and human–computer interactions, two emerging technologies have been introduced into the product assembly process—digital twins (DT) and cyber-physical systems (CPS) [3]. DT can be considered an interdisciplinary, multi-scale simulation process that makes full use of data such as sensors status, physical models, and operating history. It mirrors the physical system into a virtual space and represents the whole life cycle of the corresponding physical process. Different steps of the assembly process are connected with each other based on the communication and information technology, which reduces the deviations and as well as increases the efficiency of the entire production life cycle [4]. CPS, developed in the era of the Industrial Revolution 4.0, including network layer, perception layer, and control layer, integrates communications, control, and physical through the Internet [5]. The perception layer is mainly composed of sensors, controllers, and collectors. The sensor in the perception layer is the terminal device in the CPS and collects the needed information from the surrounding environment before sending the collected data to the application server. In the later process, the terminal device will make changes in response to the data returned from the server after processing. The data transmission layer is the connecting bridge between the information domain and the physical domain. CPS promotes the assembly systems with the ability to reconfigure process path according to real-time environment, which helps them better adapt to different product variants [6].

In order to make the robot more intelligent when dealing with complex situations in unknown environments, it is quite important to introduce the concepts of reinforcement learning and deep learning into the path planning process. Deep learning is a general term for a type of pattern analysis method that includes three types of methods: convolutional neural network (CNN), the self-encoding neural network based on multi-layer neurons, and deep belief network (DBN). In reinforcement learning, subject learns the optimal strategy in a “trial and error” mode through the interaction with the environment to obtain the greatest reward [7]. Q-Learning, one of the commonly used algorithms in reinforcement learning, takes the optimal action at a certain moment to maximize the expectation of gaining benefits. The environment will feedback corresponding rewards based on the agent’s actions. To sum up, the fundamental idea of this algorithm is selecting the action with the greatest benefit from the constructed Q-table, in which the data represents the state and action of the agent. With the application of reinforcement learning, human involvement can be reduced from the assembly system, and the robustness of the system can be increased. It also allows the assembly robot to learn

complex behaviors based on interactions with the surrounding environment. In the path planning process, it is quite important for the agent to connect with the environment and make the right decision on the next behavior, which means that the agent can perform a high-precision assembly process instead of translating human behaviors into fixed robot programs [8].

The rest of this paper is organized as follows. Section 2 reviews the related work on path planning and reinforcement learning. In Sect. 3, the digital twin framework for assembly is introduced. The detailed algorithm description and improvement of the Q-learning model are analyzed in Sect. 4. Computational experiments of the UR10 robot are conducted in Sect. 5. Finally, conclusions are given in Sect. 6, together with some discussion of potential future work.

2 Related works

2.1 Path planning

Path planning has always been a vital part of the production process. It has a great influence on production efficiency and safety. Thus, how to let the industrial robots automatically plan their path has become a very important issue. Path planning can be divided into global path planning and local path planning. The location of the obstacles in the global path planning is known. Thus, the main mission of global path planning is to build up the model of the environment after calculation and then optimize the path. Commonly used methods include the visible method, free space method, grid method, Dijkstra algorithm, and A* algorithm. However, the accuracy of global path planning is hard to ensure in a complex and changing environment. The local path planning needs the industrial robots to collect information and data from the environment through the whole assembly process, which requires the robots to be equipped with learning ability. Artificial potential field method, fuzzy control algorithm, hierarchical algorithm, neural network algorithm, and reinforcement learning method are the most commonly used methods in local path planning.

Khatib and Wu [9] first introduced the path planning of artificial potential field method, a virtual force method, to the robot field, which helps create the robot movement in the surrounding environment into an abstract artificial gravitational field. This method results in a generally smooth and safe path, while it may also lead the robot to the problem of local optimum. Zuo et al. [10] developed a fuzzy algorithm, which determined the priority of several possible paths and repeatedly drives the robot to the final configuration in the highest priority forward direction. Duguleana and Mogan [11] proposed a hierarchical path planning approach, which contains two different levels of structure. A grid-based A* algorithm was applied in the first level for finding the geometric path

and selecting multiple sub-goals path points for the next level. The least squares strategy iteration was applied into the second level to generate a smooth path under the restriction of robots' kinematics. Cruz and Yu [12] combined neural networks together with reinforcement learning into a new approach for solving the problem of robots in environments with static and dynamic obstacles. This approach helped people to take control of the robots at any speed and avoid local minima and converges in complex environments.

In recent years, researchers have also made great progress in modifying those methods above. Zhuang et al. [13] modified the classic algorithms of multi-agent so that it does not need the unvisited state. With estimating unknown environment, the paper applied the neural kernel smoothing and networks to the approximate greedy actions. Saeed et al. [14] proposed a digital twin-based assembly data management and process traceability approach for complex products. Samuel [15] generated the collision-free path with grid models and potential functions. In order to find the initial feasible path and avoid obstacles, they developed a boundary node method and a path enhancement method. The above two methods were proved to be feasible by testing in work environments in different degrees of complexity

2.2 Reinforcement learning

As one of the machine learning methods, reinforcement learning is used to describe and solve the learning strategies problem to achieve goals or maximize returns in the process of interaction between the agent and the surrounding environment. Roveda et al. [16] introduced the concept of reinforcement learning into the game of checkers and used a value function represented by a linear function to guide the selection. After years of research, the development of reinforcement learning becomes rapid, among which the Q-learning method has been widely used [16].

Recently, more and more modifications are applied to Q-learning algorithm. Konar et al. [17] provided a new deterministic Q-learning with a presumed knowledge, which used four derived properties to update the entries in the Q-table at one time. Compared with the traditional Q-learning approach, this modified method had less time complexity and smaller required storage. Robots sometimes face the problems of slow convergence speed and long-planned paths when they are in an unknown environment in applying the Q-Learning method to perform path planning. Aiming to solve these problems, Zhao et al. [18] proposed the experience-memory Q-learning (EMQL) algorithm, which continuously updated the shortest distance from the current state node to the start point. Hu et al. [19] also introduced an experience aggregative reinforcement learning method with a multi-attribute decision-making (MADM) to complete the mission of avoiding the real-time obstacle by decomposing the original obstacle avoidance task into two separate sub-tasks. Each sub-task is trained individually by a double Q-learning with a simple reward function.

3 Digital twin framework for assembly

Digital twin system integrates information control module with hardware implementation module, as shown in Fig. 1. It provides a new virtual assembly mode for industrial production, including the virtual control environment and actual physical system. The integration of computing process and physical process, in order to maintain the space–time consistency of virtual model and equipment entity, provides strong support for improving the robot path planning in an unknown environment and reducing the error between virtual and actual assembly, thus improving the application efficiency of virtual assembly in the actual industrial production. And it is more convenient for six-axis industrial robots and other new robots to be widely used in industrial production.

The sensing layer contains several measuring, feedback, and assembly devices, which have sensing, communication, and executing abilities, respectively. The sensing device, such as the grating ruler, laser sensor, and vision sensor, sends the timely measurement data to the monitor after obtaining the status attribute information concerned by the user. The execution device, including the flexible assembly part on the assembly robot as well as the whole digital production line, completes the corresponding operations according to the received instructions, such as adjusting the posture, drilling, and trimming. All collected data can be transmitted through the communication network. The ability to sense and control the physical elements is the basis of realizing the interaction with the information system.

The model library together with the perception of the underlying data constitutes the assembly scenario, including the data of each joint of the assembly robot, and the position and size of each product. Through matching and knowledge reasoning between the assembly task and scenario, the corresponding assembly process schemes are formed. The necessary assembly resources are then obtained based on the assembly scenario. The algorithm library sequentially generates the corresponding instruction code and adjustment scheme to drive the assembly equipment. During the real-time operation of the assembly equipment, the feedback equipment receives the corresponding feedback data and interacts with the simulation results. This offline pre-assembly process and CPS online simulation control further ensure the smooth assembly.

4 Theoretical framework

4.1 Q-learning algorithm

Q-learning is a value-based algorithm in reinforcement learning. Q , also represented as $Q(s, a)$, is the obtainable feedback when taking action a , under a certain state s . The main objective of this algorithm is to get the optimal Q value through iteration. A Q -table is created to reserve the Q value. The agent will select

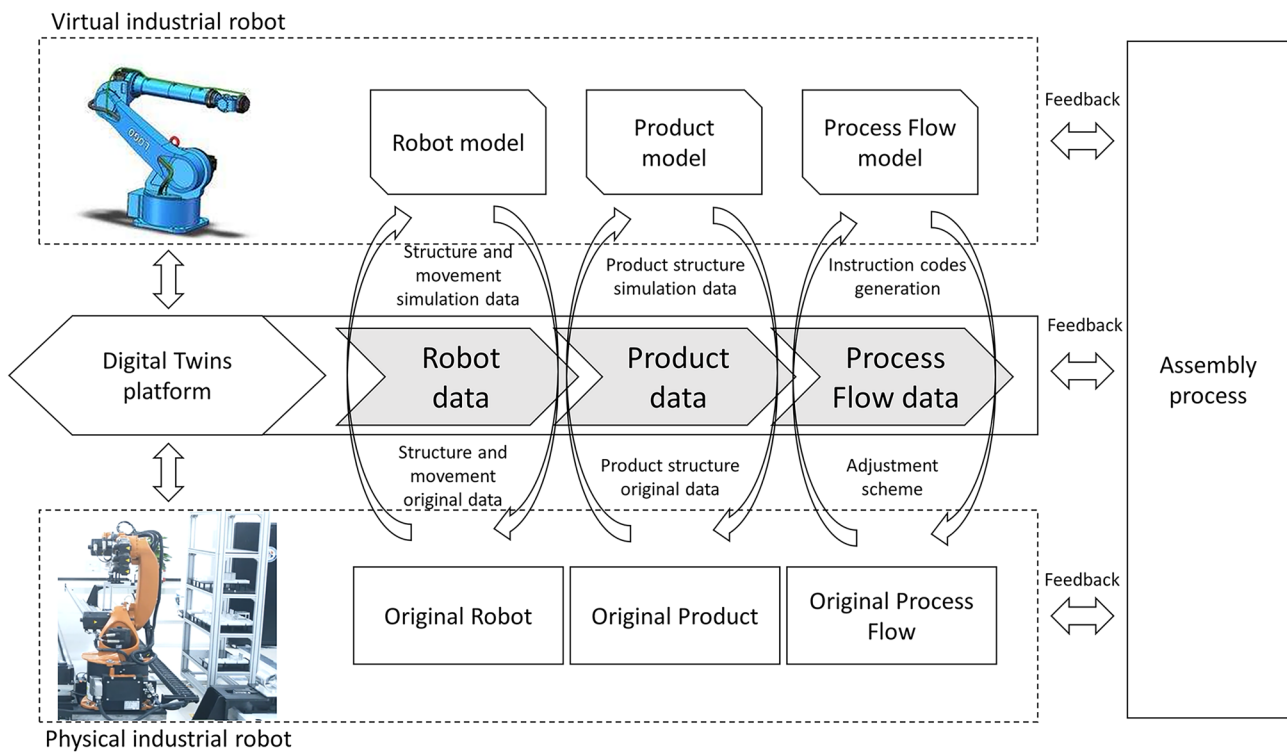


Fig. 1 Digital twin framework for assembly

an action based on the ϵ -greedy theory under a randomly given state s . After taking the given action, a new state will be generated, and a reward will be given from the outside environment. The reward value will then be reserved in a matrix R for later calculation of Q value in the next step. The Q value will then be reserved in a new matrix, called P . Finally, the action sequence will be generated based on the optimal (i.e., maximum) Q value. The detailed algorithm is shown in Table 1.

In this algorithm, s is the current state, while s' is the next state. α and a' represent the current action and next action. $Q(s, a)$ is the corresponding value of s and a ; r is the corresponding reward. γ ($0 < \gamma < 1$) is a factor that reflects the future feedback on the current situation. α represents the learning efficiency.

Table 1 Pseudocode of Q-learning algorithm

```

Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode):
  Initialize  $s$ 
  Repeat (for each step pf episode):
    Choose  $a$  from  $s$  using policy derived from  $Q$  (i.e.  $\epsilon$ -greedy)
    Take action  $a$ , observe  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'$ ;
  until  $s$  is terminal
    
```

4.2 Dual reward in the improved Q-learning algorithm

In Q-Learning, the reward function plays an important part. It helps the agent to better find the right way towards the goal. With a well-designed reward function, the agent can find the optimal way in a short time with the least steps. In most cases, the goal node is represented by an especially large reward value, while the common nodes that contain no goal and obstacles will only have a small positive number or even zero. The nodes that represent obstacles will have a negative number, which can prevent the conflict between agent and obstacles. However, with this kind of static reward function, the agent will be easily trapped in blind searching. Therefore, a dual reward function, which includes static reward function and dynamic function, is introduced in this paper.

4.2.1 The static reward function

In most cases, the state nodes can be represented by obstacles nodes, common nodes, and goal nodes. In this research, the reward function only contains two different kinds of nodes, common nodes and goal nodes. And the reward function is shown below:

$$Reward_s = \begin{cases} 0, & \text{if the node is the common node} \\ 10, & \text{if the node is the target node} \end{cases} \quad (1)$$

4.2.2 The dynamic reward function

In an unknown environment, the robot with the static reward function can be easily trapped in blind searching, for example, when the potential next state nodes of the robot are all common nodes, which means the robot will not be punished or highly rewarded no matter what action it takes in the next state. It can choose the action of any direction, or even backwards, which will increase the distance between the robot and the goal node. Therefore, the dynamic reward function is applied to calculate the distance between its current node and the target node. The dynamic reward function is shown below:

$$d_t = \sqrt{(x_{target} - x_t)^2 + (y_{target} - y_t)^2} \quad (2)$$

$$d_{t+1} = \sqrt{(x_{target} - x_{t+1})^2 + (y_{target} - y_{t+1})^2} \quad (3)$$

$$Reward_d = \lambda \frac{d_t - d_{t+1}}{|d_t - d_{t+1}|} \quad (4)$$

In these equations, d_t represents the distance between the current state node and the target node. And the d_{t+1} represents the distance between the next state node and the target node. λ is a factor that depends on static reward since it represents the proportion of dynamic reward in the total reward the agent would get. In this research, we give the value of 0.8 to λ .

4.3 Q-table optimized by case-based reasoning

In the training process of Q-learning, the initial parameter values are difficult to choose. The reward matrix in the Q-learning algorithm is a constant matrix, and the initial Q-table is often initialized to a zero-valued matrix. Therefore, it is more sensitive to the parameter values in the early stage of the algorithm and is often directly given based on empirical values. However, the same initial parameter values cannot be applied to networks of different scales and densities, so multiple experiments are required in different networks to determine the parameter values. Moreover, large space is necessary for exploration. At the beginning of the algorithm, under the guidance of a lack of prior knowledge, the agent will be trapped into the random exploration stage for a long time, and it is difficult to quickly approach the optimal solution. At the same time, random exploration may cause the algorithm to converge to the local optimal value in advance. Under the positive feedback effect of the roulette exploration strategy, the degree of exploration of the sub-optimal path is strengthened, which causes the algorithm to

fail to jump out of the local optimal. In order to make better use of the experience of historical cases, we use case-based reasoning methods to initialize the Q-table to improve the speed and quality of the training process, as shown in Fig. 2.

The case-based reasoning method has the function of analogical learning, which relies on past experience for learning and problem solving. A case library is constructed to store historical path planning cases. The solution of the new case can be obtained by modifying the case in the case library which is similar to the current scenario. First, the information used or generated in the path planning process is obtained. Each case in the case library contains information such as the task, environment, and path generated during each path planning task. By matching the context and task of the current task with the cases in the library, and evaluating and correcting the matched candidate cases, the Q table used by the case with the best performance is finally used as the initial Q table of the current task. At the same time, the case-based planning method continuously expands and updates the case library through iterative learning. In this study, the task similarity is calculated as follows:

$$S(\pi_1, \pi_2) = \frac{p(\pi_1 \cap \pi_2)}{\max(p(\pi_1), p(\pi_2))} \quad (5)$$

where π_1, π_2 respectively represent the collection of obstacle points in the two scenes. $p(\pi_1 \cap \pi_2)$ represents the number of point sets in the same area covered by obstacles in two

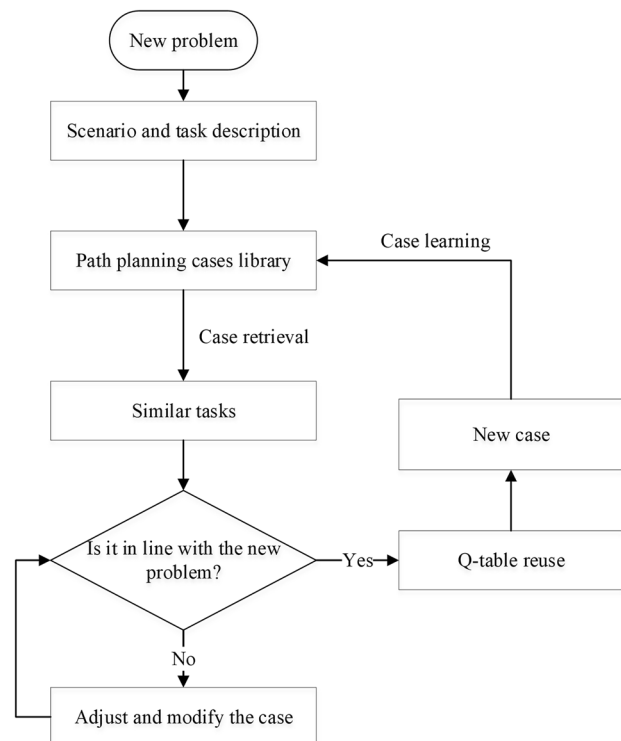


Fig. 2 Flow chart of Q-table optimization based on CBR

scenes, and $max(p(\pi_1), p(\pi_2))$ represents the number of point sets in obstacles with a larger coverage area.

4.4 Obstacle avoiding strategy

In some of the cases, a large negative reward would be set for the obstacle nodes in the reward function in order to keep the robot from encountering the obstacles. However, this research tries to record down the obstacle nodes and find the feasible actions and states at first every time the agent is going to move on to the next state. With this strategy, the possibility of hitting an obstacle can be reduced compared to the method of giving the obstacle nodes a large negative reward, since the Q-Learning method will sometimes choose actions and states randomly, ignoring the value of the reward.

5 Experiments and analysis

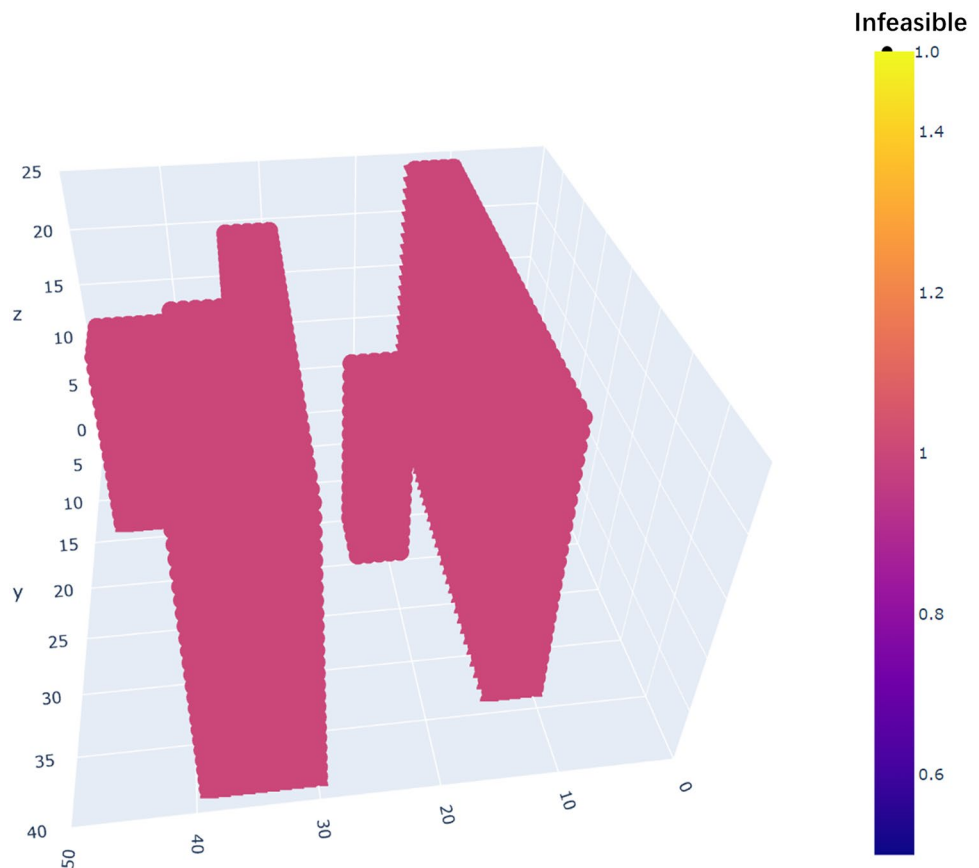
In order to verify the theoretical analysis effect of path planning based on the improved Q-learning algorithm proposed in the previous section, the manipulator and its working environment in the assembly digital twin system were modeled and verified. For specific assembly tasks, the end of the six-joint industrial robot needs to start from the starting

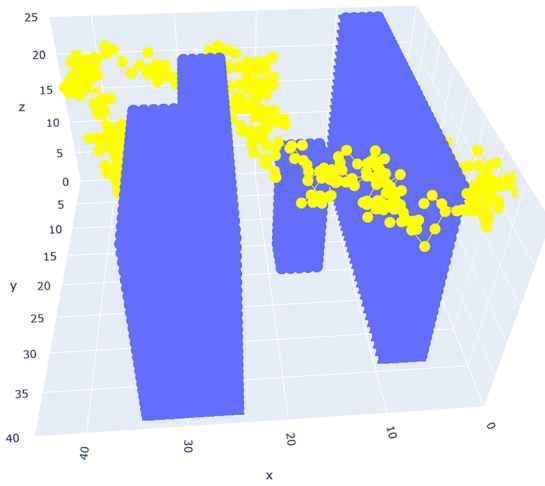
Fig. 4 Planning paths under different algorithms and iterations. **a** Original Q-learning, iteration=200; **b** original Q-learning, iteration=2000; **c** dynamic reward, iteration=200; **d** dynamic reward, iteration=2000; **e** dynamic reward and CBR, iteration=200; **f** dynamic reward and CBR, iteration=2000

position coordinate point in its working space, avoid possible obstacles in the working space, and plan the path to finally reach the target point. This study selects the path planning scenario shown in Fig. 3 below and takes a typical six-joint robot UR10 as an example to explore its performance in path-finding in the below scenario. The virtual environment contains obstacles and targets in the shape of rectangular solid. Although most of the obstacles and targets, in reality, are in different irregular shapes, we modified them into rectangular solid for calculation. The constructed virtual environment is shown below. The software and hardware environments for system implementation are summarized as follows:

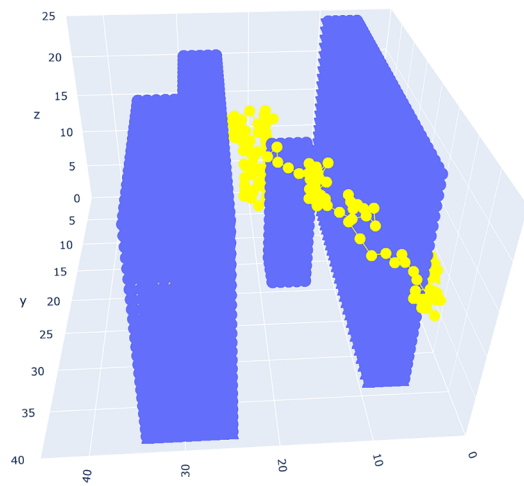
1. Application server: A ThinkPad desktop with Intel Core i7 CPU (3.40 GHz) and 32 GB memory, 2 T SCSI HD and a Windows Server operating system
2. Programming platform: PyCharm and Python 3.6.0
3. Database: MySQL 6.0
4. Network configuration: 1G Ethernet network connection between testing clients and application server

Fig. 3 Path planning environment with obstacles

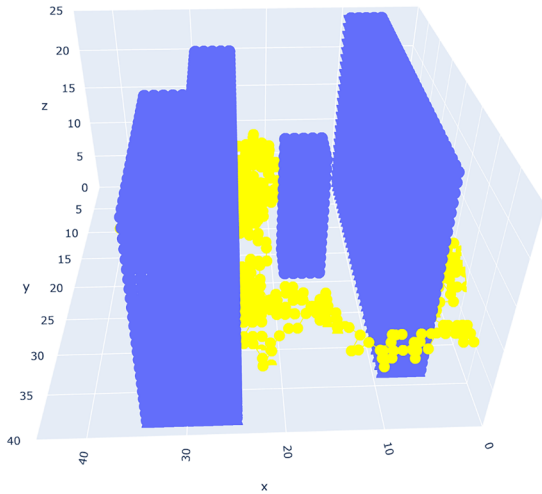




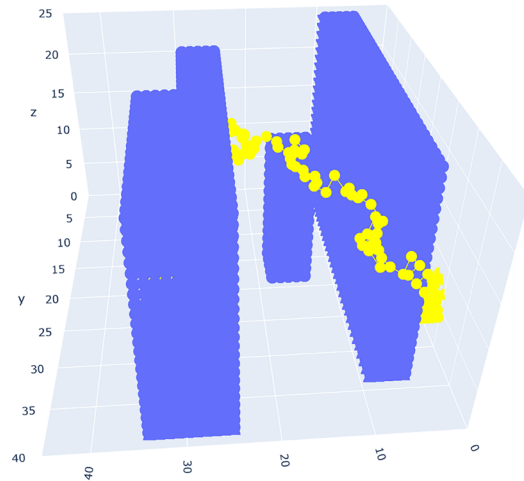
a) original Q-learning, iteration = 200



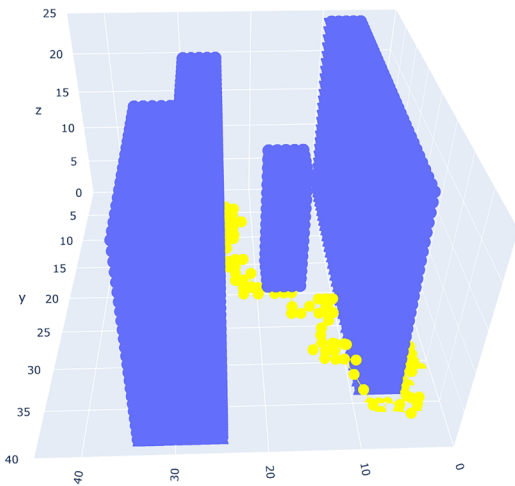
b) original Q-learning, iteration = 2000



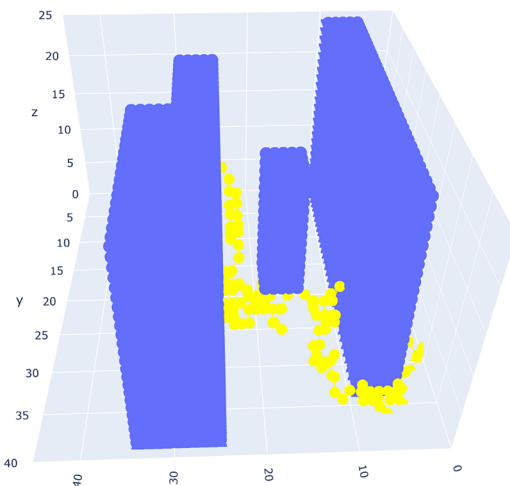
c) dynamic reward, iteration = 200



d) dynamic reward, iteration = 2000



e) dynamic reward and CBR, iteration = 200



f) dynamic reward and CBR, iteration = 2000

The actual maximum working range of the UR10 manipulator is a 1300-mm radial range. Considering the flexibility and safety of the actual assembly task of the robot, the working range is selected to be a three-dimensional space of 250mm × 200mm × 125mm. Taking 5 mm as the minimum unit for the end of the robot to move, the three-dimensional workspace of the robot can be divided into a discrete state set containing $50 \times 40 \times 25$ grids S , starting point $H(0, 0, 0)$, ending point $T(32, 30, 12)$. The pink area is the obstacles such as the workbench in the environment, and the white area is the feasible region for the robot to work freely.

The robot's possible motion directions include up, down, front, back, left, right, and a combination of the three coordinate directions. Therefore, the action set A of the robot includes basic element actions in 26 directions. By setting equal algorithm parameters, we set γ from 0 every 0.05 to 1 the maximum number of iterations to 10,000, and observe the number of steps and the number of iterations experienced when the planned path stabilizes during the process of planning the path from the start to the end of the robot.

Figure 4 shows the path of the manipulator in the initial and final stages of different algorithms. In the scene where the robot has obstacle areas, especially trap areas, using appropriate profit gain and environmental prior information can improve the efficiency of algorithm optimization, and iterative steps are fewer. Although the round update makes the update time of each step longer, after fewer rounds of update, the iterative step reduction is more advantageous, making the learning path of the improved algorithm shorter and the overall efficiency of the algorithm better. For the modified Q-Learning method in this research, dual reward function, CBR, and obtaining feasible action method are applied. Both of these methods can help the robot find the optimal route towards the target in a safer and more efficient way. For the dual reward function method, instead of the original reward function, this research combines both the static reward and dynamic reward as a whole. The former reward can help the robot identify the status of the next action, while the latter reward can lead the robot to the right direction towards the target. For the obtaining feasible action method, the robot can avoid the collision with obstacles, and the safety level of this method is much higher than that of simply setting the static reward of the obstacle nodes to a large negative value.

Furthermore, when the improved Q-learning algorithm takes three values of 0.6, 0.8, and 1 in λ , the algorithm compares and observes the steps required for each iteration of the pathfinding learning process. The index of the iteration round is taken as the horizontal axis, and the pathfinding steps of the robot in each round are taken as the vertical axis. The comparison of step changes is shown in Fig. 5. In fact, when λ is set to 0, the learning algorithm becomes the original Q-learning algorithm. Therefore, if λ is too small, the effective income of the update will be

annihilated more quickly. If λ is too large, it will cause a certain number of repeated steps to explode. Locally optimal, so the iterative effect of some rounds does not converge, and the movement is repeated in the trap area. According to the performance of different λ , we set λ to 0.8 in the following experiment.

In this study, two scenes are designed to evaluate the four algorithms, Q-learning algorithm (Q), Q-learning algorithm with CBR (CBR-Q), Q algorithm with dynamic reward function (DR-Q), and Q algorithm with dynamic reward function and CBR (D-CBR-Q). The shapes of obstacles are different in the two scenes, while the starting and ending point are the same. To compare the performance of the algorithms, three evaluation indices are utilized—descendent epoch, convergent mean steps, and the computation time. The descendent epoch denotes the epoch when the mean step in a window of 10 epochs gets 150% of the corresponding convergence step. The convergent mean step is the average step after the descendent epoch. The computation time is the time required for the robot to move from the starting point to the ending point when the algorithm converges. The descendent epoch measures the convergence efficiency, while the convergent mean step represents the optimization effect of algorithms.

Figure 6 shows the steps per epochs of different algorithms. Q, CBR-Q, DR-Q, and D-CBR-Q converge at around 6000 epochs, 4000 epochs, 2500 epochs, and 2000 epochs, respectively, when the planned path and number of iterations stabilized, and the algorithm steps decreased significantly faster. The Q-learning algorithm still does not converge when iterates 2000 epochs, and the algorithm's pathfinding performance at 2000 iterations is worse than the former at about 1500 iterations, and the pathfinding performance at 1000 iterations is the same as that of the former. From Table 2, we can see that the convergent mean steps of Q-learning in scene 1 is 423.63, CBR-Q reduces the step by 5.3%, DR-Q reduces the step by 11.2%, and D-CBR-Q reduces the step by 16.5%. As for the descendent epoch, Q-learning reaches the 150% convergence step at 517 epoch, while CBR-Q is 35.9% faster, DR-Q is 56.3% faster, and D-CBR-Q is 51.1% faster than Q-learning. The evaluation indices shows that the CBR method and dynamic reward strategy can both improves the performance of the traditional Q-learning algorithm and the proposed D-CBR-Q outperforms the other three algorithms.

In order to realize the centralized management of distributed resources by the cloud service architecture and the assembly requirements of centralized resources for decentralized services, a prototype system was developed using the web-based B/S (browser/server) architecture model. Figure 7 shows an example of editing UR10 in the equipment model library shown in the integrated Vrep modeling software.

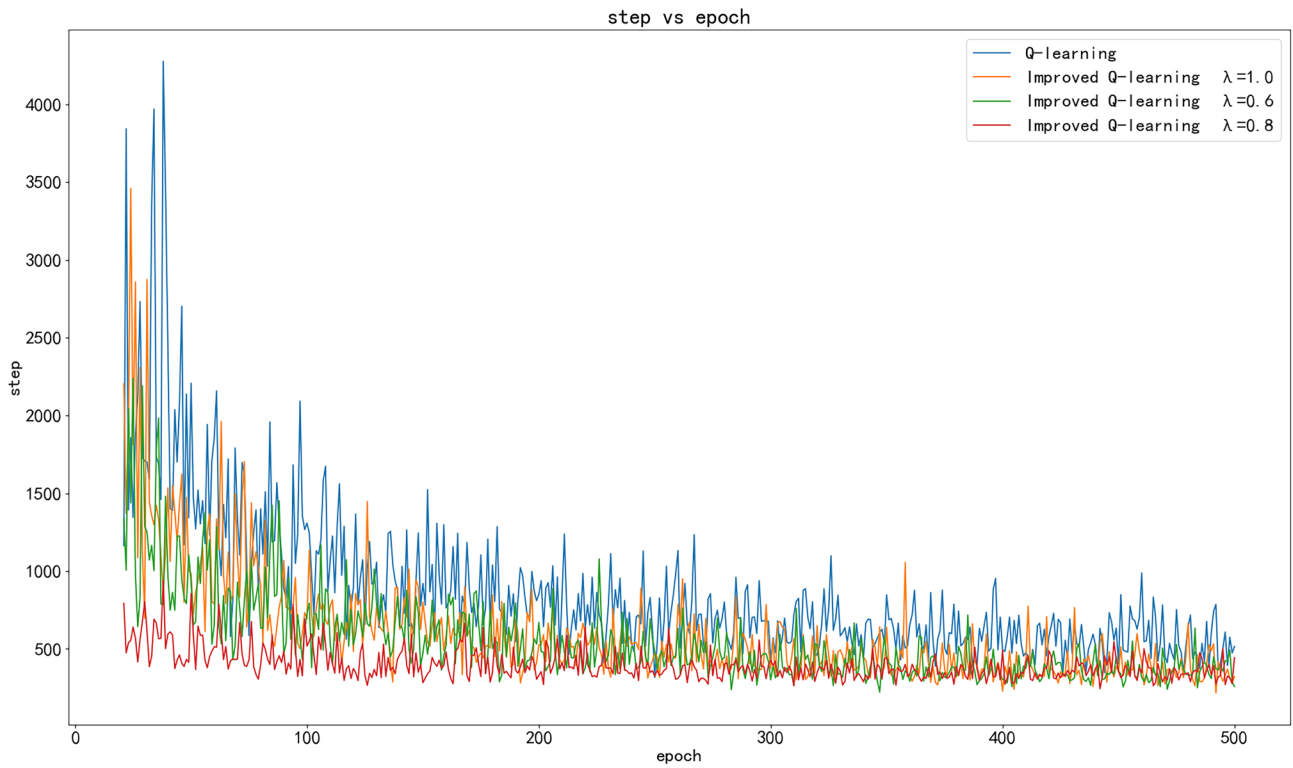


Fig. 5 Number of iterations per round under different λ

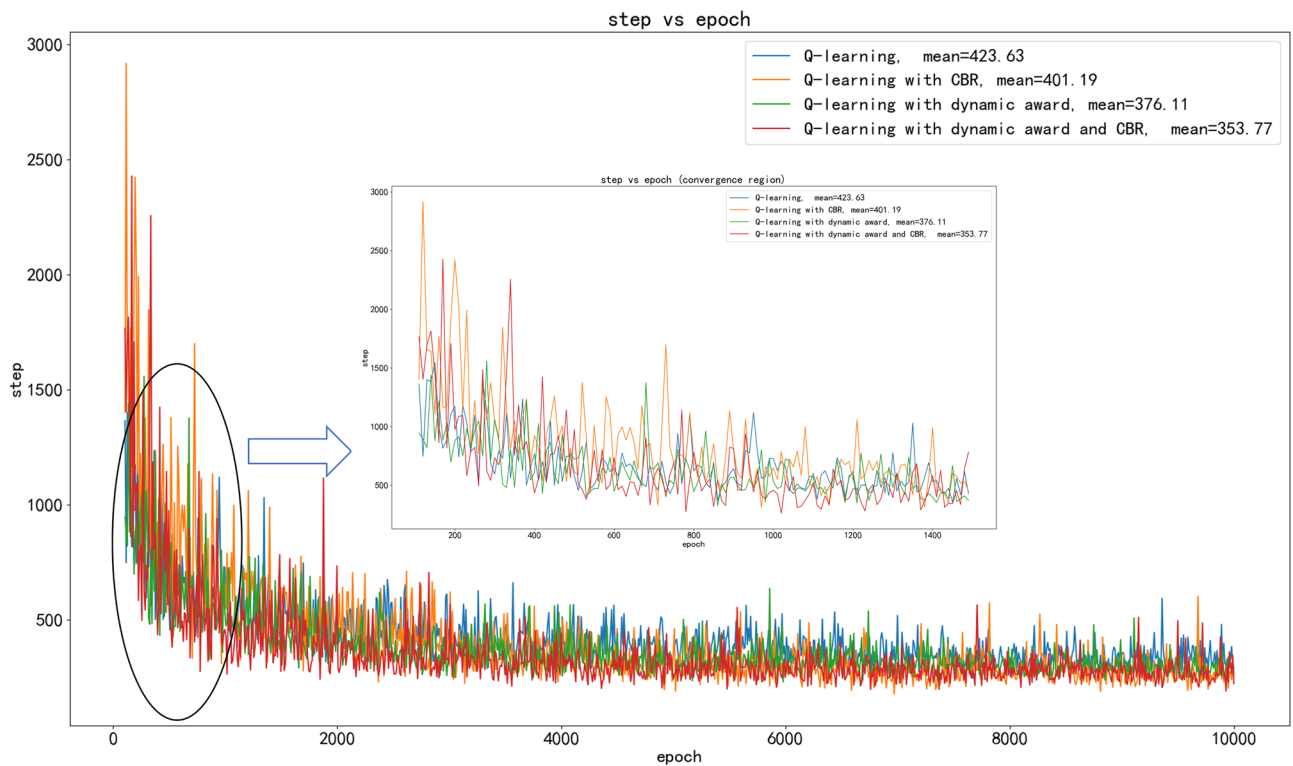
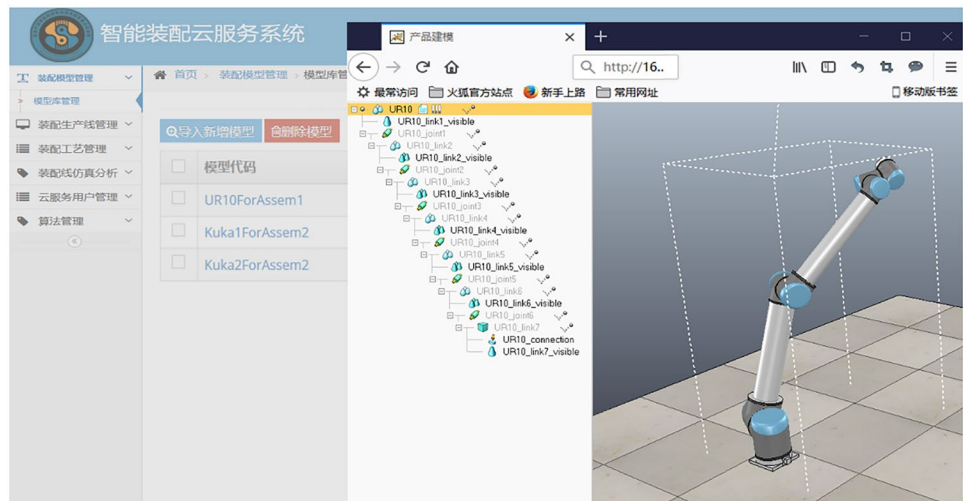


Fig. 6 Steps per epochs of different algorithms

Table 2 Comparison of different algorithms

Scene number	Algorithm	Descendent epoch	Convergent mean steps	Computation time (s)
1	Q	517	423.63	9.63
	CBR-Q	331	401.19	8.96
	DR-Q	226	376.11	6.48
	D-CBR-Q	253	353.77	4.56
2	Q	466	483.15	12.21
	CBR-Q	318	466.68	10.25
	DR-Q	207	414.58	9.07
	D-CBR-Q	194	367.85	6.10

Fig. 7 The GUI of manipulator modeling

6 Conclusions

The assembly path planning of manipulators is playing an increasingly important role in modern manufacturing. The digital twin system can provide an intelligent assembly paradigm, which realizes the real-time synchronization of virtual assembly and actual assembly through the integration of intelligent calculation process and physical process. In this research, a digital twin system for manipulator assembly is constructed, which includes functions such as real-time perception, data exchange, communication, intelligent decision-making, and virtual-real interaction. Moreover, this research also proposes a modified Q-learning algorithm, which speeds up the convergence speed by adding a dynamic reward function, optimizes the initial Q table by introducing knowledge and experience through the CBR algorithm, and prevents entry into the trapped area through the obstacle avoiding method. Finally, take the six-joint robot UR10 as an example to verify the performance of the algorithm in the three-dimensional space of pathfinding. The experimental results show that the improved Q-learning algorithm's pathfinding

performance is significantly better than the Q-learning algorithm.

As far as future research is concerned, there is great potential for the extension of the current model and solution algorithm. The proposed algorithm can be improved from the following aspects: In theoretical research, a more in-depth discussion on specific scenario modeling methods and control algorithms is still needed; application exploration also needs to be expanded from more perspectives; and the research on system architecture needs to be more detailed.

Funding This research is supported by the National Key Research and Development Plan under grant number 2020YFB1713600, the National Natural Science Foundation of China under the grant number 61903031, and the Fundamental Research Funds for the Central Universities under the grant number FRF-TP-18-035A1 and FRF-TP-20-050A2.

Availability of data and material The datasets used or analyzed during the current study are available from the corresponding author on reasonable request.

Code availability The code used during the current study is available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare no competing interests.

References

- Kumar K, Zindani D, Davim JP (Eds.) (2019) Digital manufacturing and assembly systems in industry 4.0. CRC Press
- Guo F, Cai H, Ceccarelli M, Li T, Yao B (2019) Enhanced DH: an improved convention for establishing a robot link coordinate system fixed on the joint. *Ind Robot Int J Robot Res Appl*
- Polini W, Corrado A (2020) Digital twin of composite assembly manufacturing process. *Int J Prod Res* 58(17):5238–5252
- Lu Y (2017) Cyber physical system (CPS)-based industry 4.0: a survey. *J Ind Integr Manag* 2(03):1750014
- Müller R, Vette M, Hörauf L, Speicher C (2016) Identification of assembly system configuration for cyber-physical assembly system planning. *Appl Mech Mater* 840:24–32. Trans Tech Publications Ltd
- Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT press
- Beltran-Hernandez CC, Petit D, Ramirez-Alpizar IG, Harada K (2020) Variable compliance control for robotic peg-in-hole assembly: a deep-reinforcement-learning approach. *Appl Sci* 10(19):6923
- Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. *Auton Robot Veh* 396–404. Springer, New York, NY
- Lee TL, Wu CJ (2003) Fuzzy motion planning of mobile robots in unknown environments. *J Intell Rob Syst* 37(2):177–191
- Zuo L, Guo Q, Xu X, Fu H (2015) A hierarchical path planning approach based on A* and least-squares policy iteration for mobile robots. *Neurocomputing* 170:257–266
- Duguleana M, Mogan G (2016) Neural networks based reinforcement learning for mobile robots obstacle avoidance. *Expert Syst Appl* 62:104–115
- Cruz DL, Yu W (2017) Path planning of multi-agent systems in unknown environment with neural kernel smoothing and reinforcement learning. *Neurocomputing* 233:34–42
- Zhuang C, Gong J, Liu J (2021) Digital twin-based assembly data management and process traceability for complex products. *J Manuf Syst* 58:118–131
- Saeed RA, Recupero DR, Remagnino P (2020) A boundary node method for path planning of mobile robots. *Robot Auton Syst* 123:103320
- Samuel AL (1959) Some studies in machine learning using the game of checkers. *IBM J Res Dev* 3(3):210–229
- Roveda L, Pallucca G, Pedrocchi N, Braghin F, Tosatti LM (2017) Iterative learning procedure with reinforcement for high-accuracy force tracking in robotized tasks. *IEEE Trans Industr Inf* 14(4):1753–1763
- Konar A, Chakraborty IG, Singh SJ, Jain LC, Nagar AK (2013) A deterministic improved Q-learning for path planning of a mobile robot. *IEEE Trans Syst Man Cybern Syst* 43(5):1141–1153
- Zhao M, Lu H, Yang S, Guo F (2020) The experience-memory Q-learning algorithm for robot path planning in unknown environment. *IEEE Access* 8:47824–47844
- Hu C, Ning B, Xu M, Gu Q (2020) An experience aggregative reinforcement learning with multi-attribute decision-making for obstacle avoidance of wheeled mobile robot. *IEEE Access* 8:108179–108190

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.