



Article

Optimizing Robotic Task Sequencing and Trajectory Planning on the Basis of Deep Reinforcement Learning

Xiaoting Dong ^{1,2,3,4} , Guangxi Wan ^{1,2,3,*} , Peng Zeng ^{1,2,3,*}, Chunhe Song ^{1,2,3} and Shijie Cui ^{1,2,3}

¹ State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China; songchunhe@sia.cn (C.S.); cuishijie@sia.cn (S.C.)

² Key Laboratory of Networked Control Systems, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China

³ Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China

⁴ University of Chinese Academy of Sciences, Beijing 100049, China

* Correspondence: wanguangxi@sia.cn (G.W.); zp@sia.cn (P.Z.)

Abstract: The robot task sequencing problem and trajectory planning problem are two important issues in the robotic optimization domain and are solved sequentially in two separate levels in traditional studies. This paradigm disregards the potential synergistic impact between the two problems, resulting in a local optimum solution. To address this problem, this paper formulates a co-optimization model that integrates the task sequencing problem and trajectory planning problem into a holistic problem, abbreviated as the robot TSTP problem. To solve the TSTP problem, we model the optimization process as a Markov decision process and propose a deep reinforcement learning (DRL)-based method to facilitate problem solving. To validate the proposed approach, multiple test cases are used to verify the feasibility of the TSTP model and the solving capability of the DRL method. The real-world experimental results demonstrate that the DRL method can achieve a 30.54% energy savings compared to the traditional evolution algorithm, and the computational time required by the proposed DRL method is much shorter than those of the evolutionary algorithms. In addition, when adopting the TSTP model, a 18.22% energy reduction can be achieved compared to using the sequential optimization model.



Citation: Dong, X.; Wan, G.; Zeng, P.; Song, C.; Cui, S. Optimizing Robotic Task Sequencing and Trajectory Planning on the Basis of Deep Reinforcement Learning. *Biomimetics* **2024**, *9*, 10. <https://doi.org/10.3390/biomimetics9010010>

Academic Editors: Fei Tao, Yanhua Long, Haoran Wei and Zhenghua Huang

Received: 27 November 2023

Revised: 21 December 2023

Accepted: 24 December 2023

Published: 27 December 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: robot task sequencing; trajectory planning; co-optimization; deep reinforcement learning; robotic manufacturing

1. Introduction

An increasing number of robots are being deployed on production floors to achieve intelligent manufacturing. This has motivated the development of production optimization technology in robotics. Robot task sequencing [1] and trajectory planning [2] are two traditionally separate optimization problems in robotics. The task sequencing problem focuses on finding the optimal task execution sequence for the robot in order to achieve certain objectives. It is analogous to the traveling salesman problem (TSP), but it is more complex because of the robot's kinematic redundancy (also known as the IK solution). The robot trajectory planning problem involves determining the timing of the motion law that the robot follows along a predefined geometric path while satisfying specific requirements, such as trajectory smoothness and accuracy, and achieving desired objectives, such as those related to the execution time, energy consumption, vibration, and their combinations; a general overview is given in [3].

In the robotic work cell, there exists a unique group of application scenarios that involve task sequencing and trajectory planning problems simultaneously, such as spot welding, freeform surface inspection, or spray painting [4]. In these applications, the robot is required to visit a set of task points with no predefined sequence to perform corresponding tasks, and it must finally return to its initial state. Clearly, the visiting

sequence of the task points and the trajectories followed for the robot to reach each task point have strong effects on the robot's production efficiency. Additionally, different task-point visiting sequences can lead to alterations in the robot's moving path, consequently impacting the robot's movement trajectories. Therefore, it is necessary to address the task sequencing and trajectory planning problems simultaneously.

However, the robot task sequencing and trajectory planning problems are traditionally treated as two separate issues. The robot motion trajectories are often predefined at the production control level, and then, the production scheduling level conducts the task sequence planning. This sequential optimization model diminishes the optimization space of the production problems and can improve productivity to a certain extent, but the optimization solutions are often suboptimal because of the neglect of the underlying synergies associated with the optimization objectives among the task sequencing and trajectory planning problems. In addition, the solution space for the TSTP problem grows exponentially with the number of manufacturing points. For an exhaustive search, the computational complexity is $O(m!g^m)$, where m is the number of task points and g is the number of IK solutions for each point. Traditional methods adopted to solve task sequencing and trajectory planning problems, such as metaheuristics and dispatching rules, face challenges in terms of time efficiency and ensuring high-quality solutions. These methods either suffer from slow processing times or struggle to consistently achieve optimal solutions that meet various objectives. Over the past few years, deep reinforcement learning algorithms (DRL) [5] have enabled significant advancements; their excellent self-learning and self-optimization qualities enable them to solve complex decision-making problems quickly and accurately, which has resulted in their extensive utilization across various robot work cell optimization problems [6].

Motivated by the aforementioned co-optimization requirements and DRL algorithms, in this paper, we take the optimization problems in task sequencing and trajectory planning as a monolithic problem and model it as a Markov decision process. Furthermore, a DRL-based optimization method is developed to address the above co-optimization problem. The major contributions are as follows.

- To the best of our knowledge, we are the first to combine the traditionally separate optimization problems in task sequencing and trajectory planning into a monolithic problem, called the robot TSTP problem, providing an integrated view in the discrete manufacturing domain.
- To solve the above TSTP problem, we employ a DRL-based policy for decision optimization. During the learning process, a specific state representation, action space, and reward function are carefully designed. Typically, in the action space, each action step considers the selection of task points, IK solutions, and trajectory parameters concurrently.
- Given the absence of a benchmark test for scheduling and control co-optimization, the feasibility of the proposed TSTP model and the effectiveness of the DRL are validated through a demonstration abstracted from a specific real-world case: a spot-welding task in an automation plant with a UR5 robot.

The rest of this article is organized as follows. Section 2 provides a summary of the related work. Section 3 formulates the mathematical model of the robot TSTP problem under study. Section 4 presents the methodology. Section 5 presents the experiments and results. Section 6 concludes this study.

2. Related Work

This paper describes a comprehensive study that considers the combined optimization of the robot task sequencing and trajectory planning problems. Because of the scarcity of work in this field, this section discusses the research statuses of the task sequencing and trajectory planning problems individually.

2.1. Existing Works for Robot Task Sequencing Problem

The robot task sequencing problem is essentially a tour planning problem and can be viewed as a variation of the traveling salesman problem or one of its extensions [7,8]. The formulation of the robot task sequencing problem as a mixed-integer nonlinear program problem is a common approach to its solution. However, searching for exact solutions via the mixed-integer nonlinear program method can be computationally intensive, especially for larger instances. To tackle the large-scale robot task sequencing problem, heuristics are applied, such as GA [9], PSO [10], etc. The work presented in [11] was one of the first to use GA to address the robot task sequencing problem; the authors designed an innovative encoding method that contained both the task sequence and corresponding configurations' information. In their experiments, the computational time for a 6-DoF robot with a problem size of 50 points was approximately 1800 s. Xidias et al. [12] and Zacharia et al. [13] extended this study by considering obstacles in 2D and 3D environments; they used the bump surface concept to capture obstacle information and employed a genetic algorithm to attain global solutions. However, their experiments were limited to small problem sizes, with only up to 15 target points being considered.

To reduce the computational time required for the large-scale robot task sequencing problem, the clustering algorithm and its improvement are regarded as a promising solution. Gueta et al. [14] and Wong C et al. [15] attempted to solve the robot task sequencing problem via clustering. They addressed the robot task sequencing problem in the same way, firstly dividing the task points into a predetermined number of clusters based on their topological locations. Subsequently, the robot task sequencing problem was transformed into the problem of finding a tour across clusters and determining the subsequent visiting sequences for the navigation of points within each cluster. The difference was that the former study was conducted in a joint space, while the latter was in a Cartesian space. Encouragingly, for a 6-DoF robot with one thousand task points, the clustering algorithm was able to find the solution within 200 s.

More recently, the RoboTSP algorithm [16] has emerged as another method that could compete with clustering in terms of speed. It was able to produce a high-quality solution in less than a minute for a complex drilling task involving 245 targets, with an average of 28.5 configurations per target. Hiu-Hung et al. [1] optimized the robot task sequencing problem using a two-tier model that only required seconds to obtain solutions for hundreds of points. To begin, they employed a real-coded twin-space crowding evolutionary algorithm to establish the robot configuration. Then, they optimized the task point sequence of the robot through a heuristic bidirectional reference mechanism. Despite the effectiveness of these heuristics in dealing with the robot task sequencing problem, the authors separated the optimization of task sequencing and trajectory planning. Consequently, this setup caused the performance of the downstream sequential planning to be constantly restricted by the trajectory planning results. Since robot task sequencing and trajectory planning are closely intertwined, it is imperative to optimize them concurrently in order to meet the requirements on both fronts.

2.2. Existing Works for Robot Trajectory Planning

Trajectory optimization is an important tool for the minimization of industrial robot operation times and energy consumption, which is a relevant research issue for many scholars.

Trajectory optimization is an important tool for the minimization of industrial robot operation times and energy consumption, which is a relevant research issue for many scholars. Time optimality is the earliest and most commonly used trajectory optimization index. Bobrow JE et al. [17] were the first to solve the minimum-time robot trajectory planning problem using the conventional linear feedback control theory. Lee Y D et al. [18] then attempted to use GA to solve the time-optimal trajectory planning for a two-link manipulator. With the development of intelligent algorithms, some algorithms with good performance have been introduced to address the robot trajectory planning problem, such as the DE algorithm [19], the PSO algorithm [20,21], the ant colony algorithm [22], the whale

algorithm [23], and so on. However, focusing solely on minimizing time as the objective in robot trajectory planning frequently results in significant vibrations of the robot, leading to a loss of tracking and positioning accuracy and even exacerbating motor wear. Therefore, the co-optimization of time optimality and trajectory smoothness becomes an important research area. In the work [24], a teaching–learning-based optimization algorithm was proposed to solve the optimal time–jerk trajectory planning for a 6-DoF welding robot. Zhang T et al. [25] presented a practical time-optimal and smooth trajectory planning algorithm and applied it to robot arm trajectory planning successfully. Works [26–28] adopted the NSGA-II algorithm, new convex optimization approach, and triple nurbs curves with bidirectional interpolation algorithm, respectively, to solve the optimal time–jerk trajectory planning problems for 6-DoF industrial manipulators, which greatly enriched the optimization algorithm library for robot trajectory planning.

In recent years, there has been growing concern regarding energy-efficient robot trajectory planning and the simultaneous minimization of energy and time, driven by the increasing costs of energy [29]. Various practical methods of optimizing energy along specified paths in robotic systems have been developed, such as scaling the reference trajectory, as shown in works [30,31]. Lennartson B. et al. [32] extended the time-scaling process by introducing a dynamic scaling factor to save more energy. Zhou J et al. [2] proposed a modified algorithm for B-spline feedrate curves with a callback mechanism, resulting in energy and time savings of over 45% compared to the conservative feed in the case of sculptured surface machining under complex constraints. Hou R et al. [33] formulated a time–energy optimization model using a phase plane based on the Riemann approximation method and utilized an iterative learning algorithm with neural networks to achieve an optimal trajectory for the precise control of industrial robots. W Sang et al. [34] utilized the particle swarm optimization algorithm to derive the best trajectory parameters to achieve the minimization of time and energy. Shi Xiang Dong et al. [35] focused on the dynamic optimal trajectory planning for a robot with a time–energy–jerk co-optimization objective and employed a fast and elitist genetic algorithm and multi-objective particle swarm optimization algorithm to address it.

3. Problem Formulation

The study described in this paper is regarding the simultaneous optimization of the robot task sequencing problem and trajectory planning problem. To apply optimization methods, the mathematical model of the robot TSTP problem is formulated. To enhance the understanding of the TSTP model, the mathematical models of the robot task sequencing and the trajectory planning problems are briefly introduced in advance.

3.1. The Mathematical Model of the Robot Task Sequencing Problem

For an n -DoF robot that is arranged to execute a manufacturing task with m task points $P = P_1, P_2, \dots, P_m$, the moving cost of the robot from task point l to h is denoted as $C_{l,h}$, and the robot task sequencing optimization objective can be expressed as

$$\begin{aligned}
 \min \quad & \sum_{l=1}^m \sum_{h=1, h \neq l}^m C_{l,h} x_{l,h} \\
 \text{subject to} \quad & \forall l, \sum_{l=1, l \neq h}^m x_{l,h} = 1, \\
 & \forall h, \sum_{h=1, h \neq l}^m x_{l,h} = 1 \\
 & \forall l, \forall h, y_l - y_h + m x_{l,h} = m - 1 \\
 & l \neq 1, h \neq 1, l \neq h, x_{l,h} \in 0, 1 \\
 & \text{And } \forall l, y_l = 0, y_l \in I
 \end{aligned} \tag{1}$$

where the binary variable $x_{l,h}$ is assigned a value of 1 when the path from point l to h is chosen; otherwise, it is assigned a value of 0. In addition, the dummy variables y_l and y_h are employed to ensure that all points are covered once within the selected sequence. It should be noted that in the robot TSTP problem, $C_{l,h}$ is not a constant. Rather, $C_{l,h}$ is a variable to be evaluated according to the trajectory curve after the l th and h th points are configured.

3.2. Joint Configuration Selection for the Task Points

The multiplicity of the IK solutions is one of the typical characteristics of the robot kinematics model, which means that the robot can reach the same task point in many different joint configurations; see Figure 1a. This type of multiplicity makes the robot task sequencing problem more complex but also increases the optimization potential of the robotic work cell. If a specific joint configuration is selected for the robot to reach a task point, then the task point is configured; see Figure 1b.

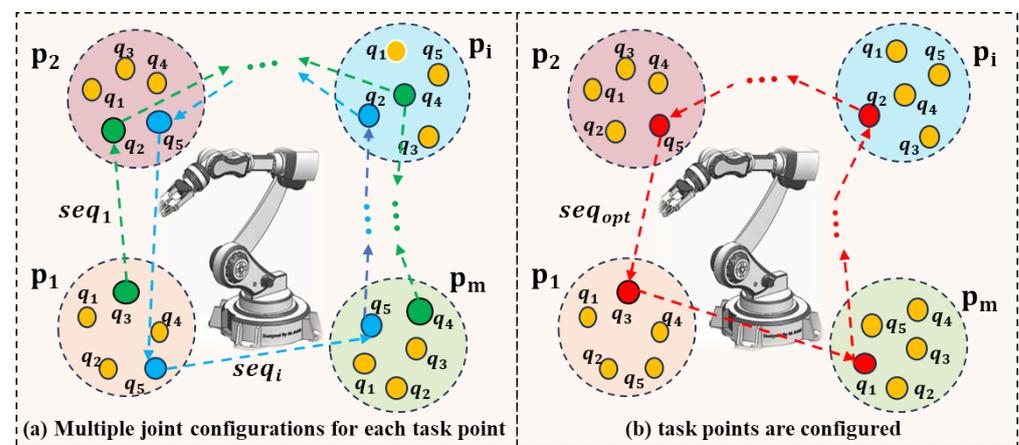


Figure 1. Joint configuration selection for task points. (a). The blue and green directed lines represent two different task points execution sequences, while the dot of the same color represent the joint configuration of the corresponding task point. (b). Example of an optimal task points execution sequence and corresponding joint configurations.

Assume that the robot needs to move from task point $P_1(x_1, y_1, z_1, Rx_1, Ry_1, Rz_1)$ to $P_2(x_2, y_2, z_2, Rx_2, Ry_2, Rz_2)$. According to the robot inverse kinematics model $q = \Gamma^{-1}(P)$, two sets of IK solutions, $Sol_1 = \{q_1, q_2, \dots, q_g\}$, $Sol_2 = \{q'_1, q'_2, \dots, q'_g\}$, can be obtained, where g is the number of IK solutions and $q_i = \{\theta_1, \theta_2, \dots, \theta_n\}$ is a group of joint configurations of the robot that can reach the task point $P_i, i \in 1, 2$. The robot transformation cost from joint configuration k to r is denoted as $t_{k,r}$, and the selection of the joint configurations for the task points can be described as follows: select a joint configuration q_k from the IK solution set Sol_1 and select another joint configuration q_r from the IK solution set Sol_2 to ensure that the $t_{k,r}$ is minimum.

3.3. The Mathematical Model of the Robot Trajectory Planning Problem

The primary objective of optimal trajectory planning in this study is to minimize the value of the robot's moving time and energy consumption simultaneously. Assume that the robot needs to move from pose $q_1 = \{\theta_1, \theta_2, \dots, \theta_n\}$ to pose $q_2 = \{\theta'_1, \theta'_2, \dots, \theta'_n\}$, with the constraints of $\dot{q}_1 = 0$, and $\ddot{q}_1 = 0$, $\dot{q}_2 = 0$ and $\ddot{q}_2 = 0$. According to the robot dynamics model, the joint torque can be calculated by

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F(\dot{q}) + G(q) \tag{2}$$

where q , \dot{q} , and \ddot{q} are the vectors of the joint angles, joint velocities, and joint accelerations, respectively. $M(q)$ is the mass matrix, $C(q, \dot{q})$ is the Coriolis and centripetal coupling matrix, $F(\dot{q})$ is the friction force vector (comprising both viscous friction and Coulomb

friction), $G(q)$ is the gravity vector, and τ is the joint torque vector. Thus, the energy consumption can be articulated as

$$\begin{aligned}
 E &= \int_0^T |\tau \cdot \dot{q}| dt \\
 \text{subject to } \tau_j^{min} &\leq |\tau_j| \leq \tau_j^{max} \\
 \dot{q}_j^{min} &\leq |\dot{q}_j| \leq \dot{q}_j^{max}
 \end{aligned} \tag{3}$$

where τ_j^{min} and τ_j^{max} and \dot{q}_j^{min} and \dot{q}_j^{max} are the lower and upper bounds of the torque and velocity of the j th joint, respectively. Then, a model for optimal time–energy trajectory planning is developed within the joint space, described as

$$\min \alpha_1 T + \alpha_2 E \tag{4}$$

where T is the moving time for the robot from pose q_1 to q_2 . $\{\alpha_1, \alpha_2\} \in R^+$ ensures that T and E are in the same order of magnitude. Note that, in the robot TSTP problem, the moving time T is a variable that is determined by the IK solution selection and trajectory curve quality.

3.4. The Mathematical Model of the Robot TSTP Problem

The above analysis shows that the robot moving cost is closely related to the moving trajectories, which are further influenced by the task point execution sequence and joint configuration selection when a specific task is executed. Traditionally, the robot trajectory planning problem is addressed at the production control level, while the robot task sequencing problem is addressed at the scheduling level. This type of sequential optimization model ignores the underlying synergies associated with the optimization objectives among the task sequencing and trajectory planning problems and cannot obtain truly global optimal solutions. To cope with this problem, we construct a co-optimization model that integrates the robot task sequencing and trajectory planning problems as a holistic problem, called the robot TSTP model.

The robot TSTP problem can be described as follows. An n -DoF robot manipulator is arranged to execute a manufacturing task with m task points $P = \{P_1, P_2, \dots, P_m\}$ with no predefined order. Each task point has six attributes $P_i = (x_i, y_i, z_i, Rx_i, Ry_i, Rz_i)$ to define its position and orientation. The robot TSTP problem seeks to simultaneously plan a set of trajectories $traj_{opt} = (traj_0, traj_1, \dots, traj_{m-1})$ and find a task point execute sequence $P_{opt} = \{P_x, P_y, \dots, P_m, \dots, P_k\}$ to achieve time and energy consumption minimization; see Figure 2. This can be formulated as

$$\min C_{P_0, P_x} + C_{P_x, P_y} + \dots + C_{P_i, P_{i+1}} + \dots + C_{P_k, P_0} \tag{5}$$

$$C_{P_i, P_{i+1}} = \alpha_1 T_i + \alpha_2 E_i \tag{6}$$

where $C_{P_i, P_{i+1}}$ represents the moving cost of the robot from task point P_i to P_{i+1} and T_i and E_i are the time and energy consumption correspondingly. P_0 is the robot’s initial state. $\{\alpha_1, \alpha_2\} \in R^+$ ensure that T_i and E_i are in the same order of magnitude. It should be noted that T_i and E_i are variables evaluated according to the trajectory $traj_i$ between task points P_i and P_{i+1} . Assuming that the robot moves from P_i using the l th configuration to P_{i+1} using the r th configuration, the smallest moving time $T_{i,min}$ can be calculated by the following:

$$T_{i,min} = \max\left(\frac{|\theta_j^{i+1,r} - \theta_j^{i,l}|}{\dot{\theta}_{jmax}}\right), j = 0, 1, \dots, n \tag{7}$$

In Equation (7), the robot is always required to move at its maximum speed. Although this can reduce the robot’s moving time significantly, it can also lead to increased wear and tear on the robot’s components and higher energy consumption. To balance the robot’s

moving speed and energy consumption, a set of time factors $\omega = \{1.0, 1.05, 1.1, 1.15, 1.2\}$ is designed to make the robot moving time T_i variable and controllable. In this case, the robot moving time T_i can be calculated by the following:

$$T_i = T_{i,min} * \omega_i \tag{8}$$

where $\omega_i \in \omega$, which is also the control parameter of the trajectory planning that should be optimized together with the robot task points sequence in our Co-RTSATP problem.

$$\tau_i = M(q_i)\ddot{q}_i + C(q_i, \dot{q}_i)\dot{q}_i + F(\dot{q}_i) + G(q_i) \tag{9}$$

$$E_i = \int_{T_0}^{T_i} |\tau_i \cdot \dot{q}_i| dt$$

$$\text{subject to } \tau_j^{min} \leq |\tau_{i,j}| \leq \tau_j^{max} \tag{10}$$

$$\dot{q}_j^{min} \leq |\dot{q}_{i,j}| \leq \dot{q}_j^{max}$$

τ_i is a vector of the joint torque; it is a variable that varies from τ_{P_i} to $\tau_{P_{i+1}}$ along with time. $\tau_{i,j}$ and $\dot{q}_{i,j}$ are the torque and velocity, respectively, of the j th joint when moving from task point P_i to P_{i+1} .

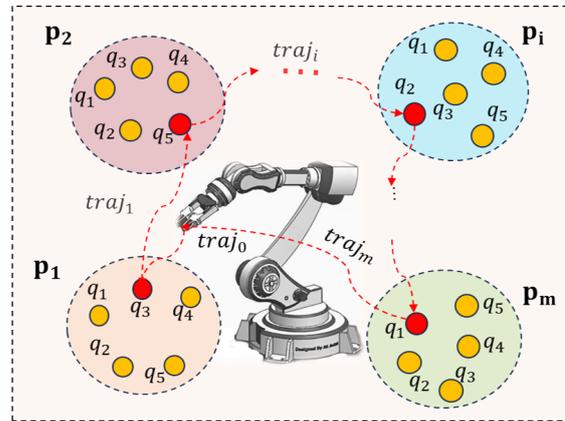


Figure 2. Illustration of the robot TSTP problem. The red directed curves and dots represent the optimal task points execution sequence and corresponding robot moving trajectories between task points.

3.5. Assumptions

In developing the methodology to optimize the robot TSTP problem, the following requirements must be satisfied:

- The robot needs to pass through each task point only once and return to its initial pose after completing all task points;
- No obstacles should exist between the task points;
- After reaching a task point, the robot should move directly to the next task point; the time taken to finish the task is not considered when calculating the cycle time because the time required to finish a task depends only on a specific process;
- The robot kinematics and dynamics should be known.

4. Methodology

In this section, the robot trajectories between task points are planned using the quintic polynomial interpolation (QPI) algorithm, and the global optimization process of the robot TSTP problem is implemented via the DRL algorithm.

4.1. The QPI Algorithm

The QPI algorithm is widely used in the realm of robot trajectory planning because of the smoothness and lack of acceleration jumps. In the robot TSTP problem, the general mathematical expression of the QPI algorithm is

$$\begin{cases} q(t) = a_1 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5 \\ \dot{q}(t) = a_1 + 2a_2t + 3a_3t^2 + 4a_4t^3 + 5a_5t^4 \\ \ddot{q}(t) = 2a_2 + 6a_3t + 12a_4t^2 + 20a_5t^3 \end{cases} \quad (11)$$

where $a_0, a_1, a_2, a_3, a_4, a_5$ satisfy:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}^{-1} \begin{bmatrix} q(0) \\ \dot{q}(0) \\ \ddot{q}(0) \\ q(f) \\ \dot{q}(f) \\ \ddot{q}(f) \end{bmatrix} \quad (12)$$

According to Equation (11), the quintic polynomial coefficient matrix $A = [a_0, \dots, a_5]^T$ can be calculated when the joints' angle positions, velocities, and accelerations are determined and the time t is given. In the robot TSTP problem, the joints' angles, velocities, and accelerations at the start and end points are known, as shown in Formula (12), and T_i is obtained by the DQN. Thus, A can be solved.

$$\begin{cases} t_0 = 0, t_f = T_i \\ q(t_0) = q_i, q(t_f) = q_{i+1} \\ \dot{q}(t_0) = 0, \dot{q}(t_f) = 0 \\ \ddot{q}(t_0) = 0, \ddot{q}(t_f) = 0 \end{cases} \quad (13)$$

An example of using QPI to plan the robot moving trajectories is shown in Figure 3, where $q_j(0) = 1.81$ rad, $q_j(2.3) = 4.62$ rad, $\dot{q}_j(0) = 0$ rad/s, $\dot{q}_j(2.3) = 0$ rad/s, $\ddot{q}_j(0) = 0$ rad/s², and $\ddot{q}_j(2.3) = 0$ rad/s². The curves of the joint angle, angular velocity, and angular acceleration are smooth, with no mechanical vibration, which is beneficial for the robot's service life.

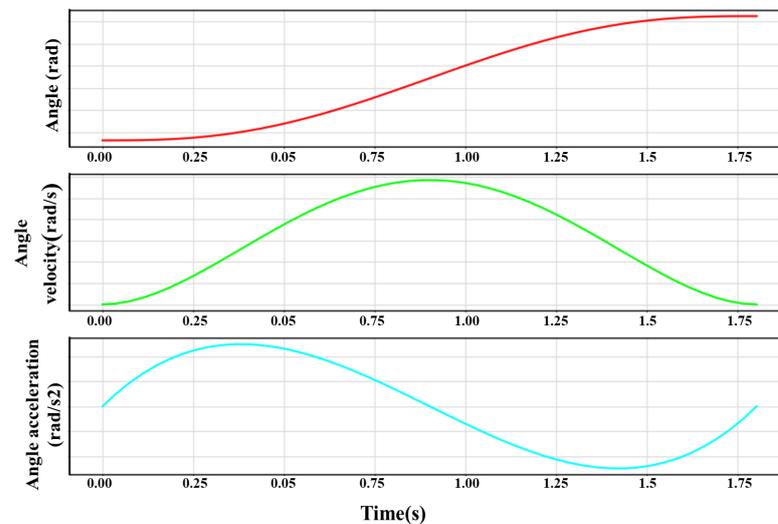


Figure 3. An example of robot trajectory planning by the QPI algorithm.

4.2. DRL Algorithm

The deep Q-learning neural network algorithm (DQN) is a type of DRL algorithm that integrates the RL techniques and supervised learning [36]. The proposed DQN framework for the robot TSTP problem is illustrated in Figure 4.

The DQN algorithm utilizes two deep neural networks: the evaluation network Q and the target network \hat{Q} . First, the agent observes the initial state s_0 of the robot TSTP problem environment and computes the state features' values. Then, the agent takes an action to interact with the environment using the $\epsilon - Greedy$ policy. Next, the agent receives a reward from the environment, and the Q -network is trained through gradient descent based on the calculated error. Finally, a new state s' and its corresponding state features' values are computed before the next decision point. In this way, the agent constantly interacts with the environment and records its experiences (s, a, r, s') within the environment. The reward (r) received for the action (a) in a given state (s) and the resulting new state (s') after the action are stored in the experience replay memory. When a sufficient number of experiences has been accumulated, an amount equal to the size of the mini-batch is randomly selected from the experience replay memory to train the Q -network. As the learning progresses, the experience is constantly updated. By reusing and learning from past experiences, the agent can learn from a more diverse and robust set of experiences, which, in turn, helps to decorrelate the data samples and break the temporal correlations in the sequence of experiences. This is important to prevent the learning process from falling into local optima and to make the learning process more stable and data-efficient.

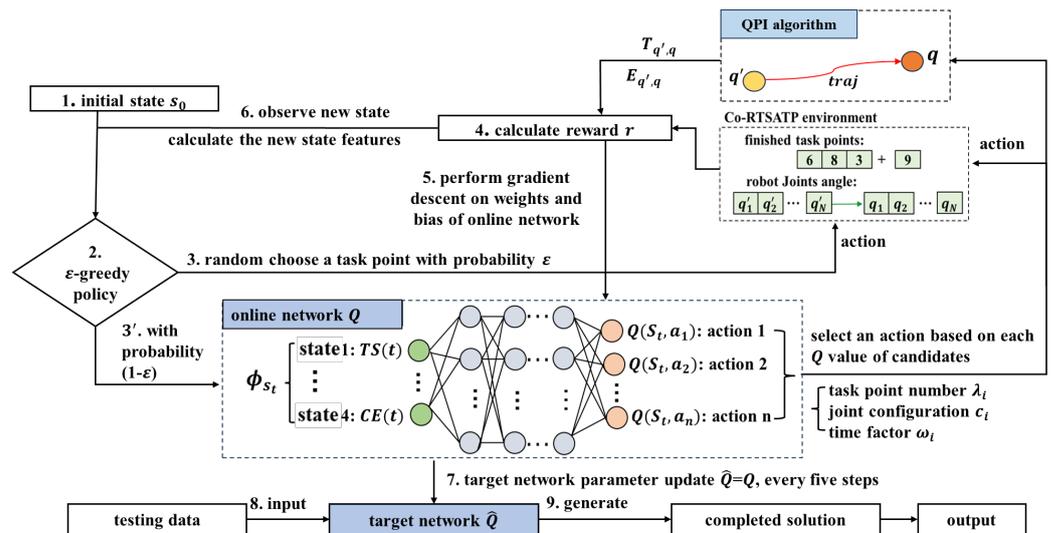


Figure 4. Algorithm framework for the robot TSTP problem.

4.2.1. Solution Representation

In the DQN algorithm, the solution of the robot TSTP problem can be expressed by the task points sequence vector, joint configuration number vector, and time factor vector; each vector contains m elements. In the task points sequence vector $\{\lambda_1, \lambda_2, \dots, \lambda_m\}$, element $\lambda_i \in [1, m]$ is described by a task point number. In the joint configuration vector $\{c_1, c_2, \dots, c_m\}$, element $c_i \in [1, g]$ is represented by an IK solution number. In the time factor vector $\{\omega_1, \omega_2, \dots, \omega_m\}$, element $\omega_i \in \{1.00, 1.05, 1.10, 1.15, 1.20\}$. A solution representation is illustrated in Figure 5, where the red parts connected by the green arrows indicate that the robot reaches task point 9 with the 7th joint configuration, the moving time from the last task point 5 to point 9 is equal to $T_{5,min} \times 1.05$, and the same applies for other task points. Note that the last element marked in blue represents the time factor taken by the robot to move from the initial pose to the first task point 3, as well as the time factor from the last task point 6 back to the initial pose.

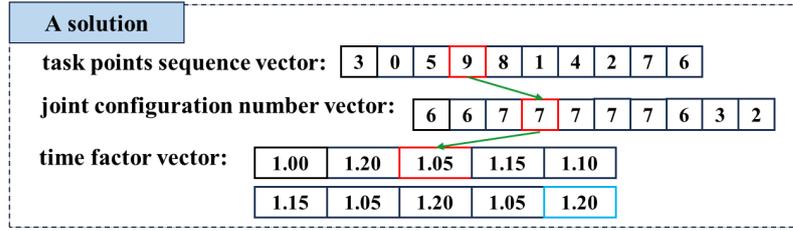


Figure 5. Solution representation in the DQN algorithm.

4.2.2. Decoding

The fitness of the robot TSTP problem can be calculated when a corresponding solution is provided. An example of the encoding process for the solution from Figure 5 is explained as follows. The robot first selects task point 3 for processing. According to the pose information $P_3 = (x_3, y_3, z_3, Rx_3, Ry_3, Rz_3)$ and the inverse kinematic model $q = \Gamma^{-1}(P)$, a set of IK solutions $Sol = \{q_1, q_2, \dots, q_8\}$ is formed. Each IK solution q_i represents a type of joint configuration used to reach task point 3, and the 6th joint configuration s_6 is chosen for the robot to reach task point 3 according to the joint configuration number vector. Because this is the first task point, the robot needs to move from the initial pose q_0 to q_3^6 by the time $max(\frac{|\theta_j^{9,7} - \theta_j^{5,7}|}{\dot{\theta}_{jmax}}) \times 1.05$ according to the time factor vector. Next, we plan the moving trajectory $traj_{0 \rightarrow 3}$ using the QPI algorithm between the initial pose q_0 and pose q_3^6 . Once the trajectory expression is known, the moving time $T_{0 \rightarrow 3}$ and energy consumption $E_{0 \rightarrow 3}$ can be calculated. By analogy, after all the task points in the solution are calculated, the total time and energy consumption of the robot in executing the manufacturing task can be obtained by summation.

4.2.3. State Features

Good state features should strike a balance between capturing environmental information accurately and avoiding unnecessary computational overhead. Based on the manufacturing task characteristics and optimization objective, four state features are designed as follows.

(1) Task state $TS(t)$

$$TS(t) = (I_1(t), I_2(t), \dots, I_m(t)) \tag{14}$$

$TS(t)$ is a binary vector whose length is equal to the task points number m , and $I_i(t) = 1$ indicates that task point p_i has been executed at decision point t ; otherwise, it has not.

(2) Robot joint state $JS(t)$

$$JS(t) = (\theta_1(t), \theta_2(t), \dots, \theta_n(t)) \tag{15}$$

$JS(t)$ is an angle vector whose length is equal to the robot joint number n , and $\theta_i(t)$ is the current angle of the i th joint. For example, $\theta_2(t) = 2.5$ rad indicates that the angle of the 2nd joint is 2.5 rad at decision point t .

(3) Current total moving time $CT(t)$

$$CT(t) = \sum_{i=1}^{\kappa} T_i \tag{16}$$

$CT(t)$ is a cumulative variable that is equal to the sum of the moving time spent by the robot at decision point t . κ represents the last completed task point P_{κ} at decision point t .

(4) Current total energy consumption $CE(t)$

$$CE(t) = \sum_{i=1}^{\kappa} E_i \quad (17)$$

$CE(t)$ is a cumulative variable that is equal to the sum of energy consumed by the robot at decision point t . κ represents the last executed task point P_{κ} at decision point t .

4.2.4. Actions

Practically, the agent has three decision variables at each decision point t . They are as follows:

- The task point number λ_i to be executed next;
- The joints configuration number c_i for the robot to reach the task point p_{λ_i} ;
- The time factor ω_i for the robot to move from the current pose to task point p_{λ_i} .

Thus, an action can be expressed by a decision vector $\mathbf{a}(t) = (\lambda_i, c_i, \omega_i)$, where $\lambda_i \in [1, m]$, $c_i \in [1, g]$ and $\omega_i \in \{1.00, 1.05, 1.10, 1.15, 1.20\}$. Algorithm 1 illustrates the procedure for the implementation of actions at each decision point.

Algorithm 1 Action implementation

- 1: **input** : probability ϵ , all output nodes Q – values $\in DQN$
 - 2: **output** : an action $a = (\lambda_i, c_i, \omega_i)$
 - 3: generate a random number ξ
 - 4: **if** $\xi \leq \epsilon$, **then**
 - 5: select the action whose Q – value is the largest
 - 6: **else**
 - 7: select an action randomly
 - 8: **end**
-

4.2.5. Reward Function

After action $a_{t+1} = (\lambda'_i, c'_i, \omega'_i)$ is implemented, the agent can observe at least one of the following three types of environment state features and obtain a corresponding reward. The specific reward calculation procedure is illustrated in Algorithm 2.

- The task point $p_{\lambda'_i}$ has been completed, and the agent receives a substantial negative reward.
- The agent completes its target, $TS_{t+1} = TS_{end}$, and the agent receives a substantial positive reward.
- The agent completes the task point $p_{\lambda'_i}$ and obtains a normal reward that relates to the time and energy increment.

Algorithm 2 Reward definition

- 1: **input** : CT and CE at decision point t and $t + 1$
 - 2: $TS(t) = (I_1(t), I_2(t), \dots, I_m(t))$ at decision point t
 - 3: $TS(t + 1) = (I'_1(t + 1), I'_2(t + 1), \dots, I'_m(t + 1))$ at decision point $t + 1$
 - 4: action $a_{t+1} = (\lambda'_i, c'_i, \omega'_i)$ at decision point $t + 1$
 - 5: **output** : reward $_{r+1}$
 - 6: **if** $I(t + 1)_{\lambda'_i} = 1$
 - 7: reward $_{r+1} = -300$
 - 8: **else if** $TS(t + 1) = TS_{end}$
 - 9: reward $_{r+1} = 10$
 - 10: **else**
 - 11: $\Delta CT = CT(t + 1) - CT(t)$
 - 12: $\Delta CE = CE(t + 1) - CE(t)$
 - 13: reward $_{r+1} = -(w_1 \Delta CT + w_2 \Delta CE)$
 - 14: **end**
-

4.2.6. DQN Topology and Training

The DQN algorithm proposed in this study consists of one input layer, three fully connected hidden layers, and one output layer. The number of nodes in the input and output layers matches the state feature and action numbers. Rectified linear unit (ReLU) activation functions are used in the hidden layers. The online network Q_e is trained through the stochastic gradient method *RMSprop* according to the error $Error = \frac{1}{|A|} \sum_{i=1}^{|A|} (\hat{Q}_i - Q_i)^2$, while the target network \hat{Q} is updated by $\hat{Q} = Q$ every five steps.

5. Experiments and Results

Robots can be programmed to perform repetitive and time-consuming tasks; when the number of task points is large, even small improvements in the production system can yield substantial benefits. Therefore, solving the robot TSTP problem for actual robotic work cells is very valuable.

The proposed approach was tested with four sets of experiments. The experimental data were randomly generated based on a spot-welding task in a real automation plant with a UR5 robot. Actually, any six-axis articulated robot is suitable for this research. All experiments were implemented in Python 3.7 on an Intel Core i7 3.0-GHz computer with 10 GB RAM and Windows 10 OS.

5.1. Robot Model

The UR5, which is the specific robot used for the experiments, is a 6-DoF robot that only has rotational joints. The standard D-H parameters derived from the Python robotics toolbox are presented in Table 1. Since the UR5 is manipulated by intersecting its last three axes at a point, only eight IK solutions are available for each task point, i.e., $Sol = \{q_1, q_2, \dots, q_8\}$.

Table 1. Denavit–Hartenberg (DH) parameters of the UR5.

q_i	q_i	a_i	α_i
q_1	0.08946	0	π
q_2	0	−0.425	0
q_3	0	−0.3922	0
q_4	0.1091	0	π
q_5	0.09465	0	− π
q_6	0.0823	0	0

5.2. Parameter Setting

In the DQN algorithm, three key parameters, ϵ in ϵ -greedy policy, reward decay γ , and learning rate l , are calibrated. The levels of the three parameters are $\epsilon \in \{0.80, 0.85, 0.90, 0.95\}$, $\gamma \in \{0.5, 0.7, 0.9\}$, and $l \in \{0.001, 0.005, 0.01, 0.015, 0.02\}$. The three parameters have $4 * 3 * 5 = 60$ combinations. Calibration experiments are conducted on the instance of 10 task points with all combinations. The result in each combination is the average fitness in 10 repeats. Figure 6 illustrates the factors' level trends of the above three parameters. According to Figure 6, we set $\epsilon = 0.9$, $\gamma = 0.9$, $l = 0.005$, and the other parameters are as follows: episode number $L = 2000$, memory size $M = 300$, and mini batch size $min_{batch} = 64$. In addition, parameters α_1 and α_2 in optimization objective Equation (6) are set to 0.2 and 0.01, respectively, based on the calibration experiment results.

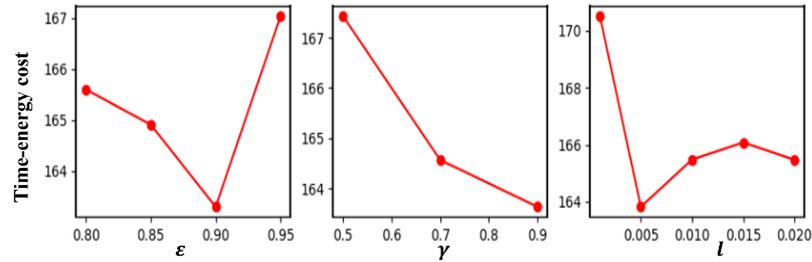


Figure 6. Factors’ level trends of parameters in the DQN algorithm.

5.3. DQN Performance Testing

With the above parameter settings, we test the global optimization performance and convergence of the DQN algorithm based on the instance of 10 task points. In this case, the comprehensive time–energy consumption cost is 118.43, in which the cycle time and energy consumption are 18.38 s and 351.87 J, respectively. When solving the robot TSTP problem, the DQN algorithm shows an excellent global optimization ability; it is always able to continue learning during the decision process. As shown in Figure 7a, in the first 50 episodes, the agent constantly interacts with the environment to accumulate experience and optimize its decisions, and its decision-making improves rapidly. Then, the accumulated experience of interacting with the environment gradually becomes saturated, the decision-making ability improves steadily, and it finally finds the optimal solution at 1209 episodes. With the progression of the learning process, the loss function value gradually decreases to zero; see Figure 7b. This shows that the DQN algorithm has good convergence when solving the robot TSTP problem.

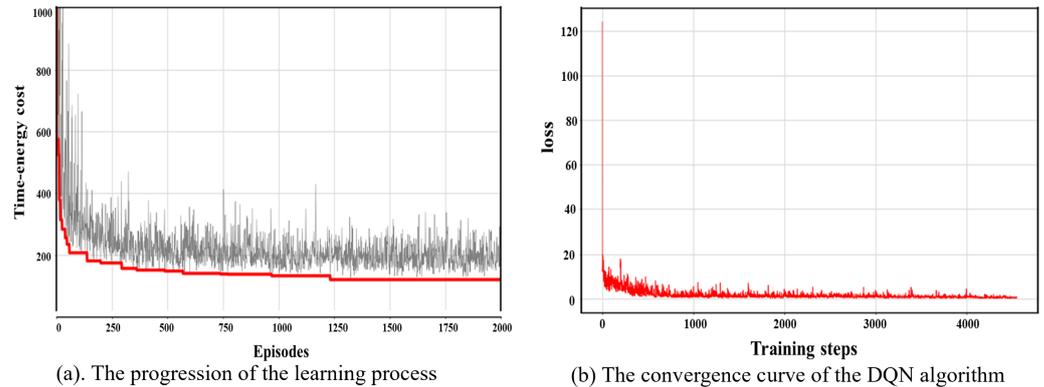


Figure 7. Decision-making process of the DQN algorithm when solving the Co-RTSATP problem.

5.4. DQN Algorithm Solves Large-Scale TSTP Problems

To test the DQN algorithm’s performance in solving large problems, we conducted experiments with different problem sizes. Moreover, another two groups of experiments whose optimization objectives are only time and only energy are conducted to verify the rationality of taking time–energy consumption as an optimization objective in the robot TSTP problem. Each experiment was conducted 10 times, and the average results were collected, as shown in Figure 8. It is obvious that the DQN algorithm could solve the robot TSTP problem with different sizes efficiently. When taking the time–energy consumption as the optimization objective, the agents could always find a compromise solution in which the cycle time and energy consumption were both slightly larger than those considering cycle time or energy consumption optimization only, but they could save time or energy significantly. For example, for 100 task points, the moving time obtained by the time–energy cooperation model is 5% higher than the time-optimal model, but the energy consumption is reduced by 35%. Likewise, compared with the energy-optimal model, the energy consumption obtained by the time–energy cooperation model is 4.01% higher, but the moving time is reduced by 25.47%. As is well known, production is a

complex process of multi-resource integration, and production efficiency can be affected by many factors, which are always in conflict with each other, such as the robot moving time and energy consumption. Therefore, this type of compromise solution of multi-objective optimization is more closely aligned with the actual requirements of manufacturing.

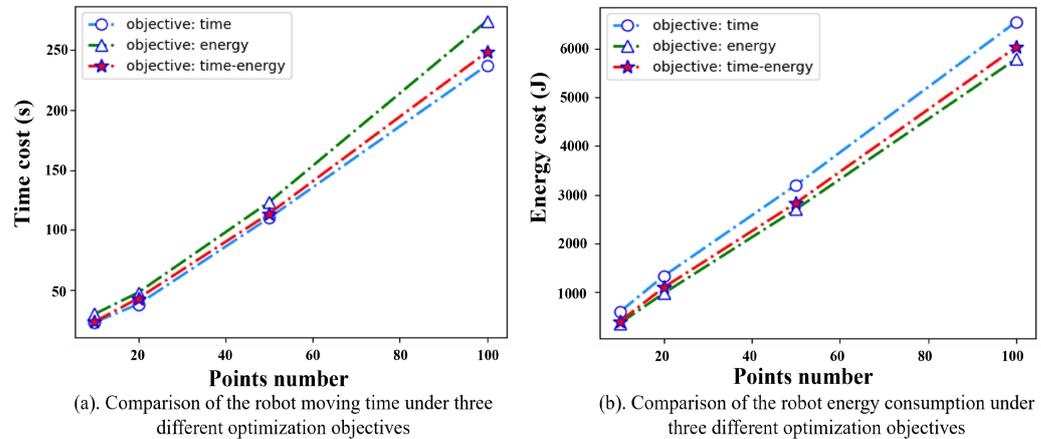


Figure 8. Results comparison of using DQN algorithm performance under different problem sizes.

5.5. Comparison with Sequential Optimization Model

Comparing studies that address similar problems is challenging. This complexity stems from variations in the approaches used to tackle the problem, as well as the considerations of robotic constraints and the specific industrial environment. Additionally, there is no universal reference for the robot TSTP problem (refer to [4] for benchmarks and evaluation limitations). Despite these constraints, we conducted a comparison of our TSTP model with [37] (abbreviated as Seq-opt model) by applying its principle to the same problem, as they only considered the robot task sequencing problem under the condition that the robot always moves at its maximum joint speed between all task points.

Figure 9a shows that the robot moving time obtained by the Seq-opt model is comparable to that of the TSTP model or even shorter. However, the corresponding energy consumption is much higher than that of the TSTP model; see Figure 9b. For example, in the experiment with 50 task points, the robot moving time obtained by the Seq-opt model is only 3.58% shorter than that of the TSTP model, but the energy consumption increases by 23.71%.

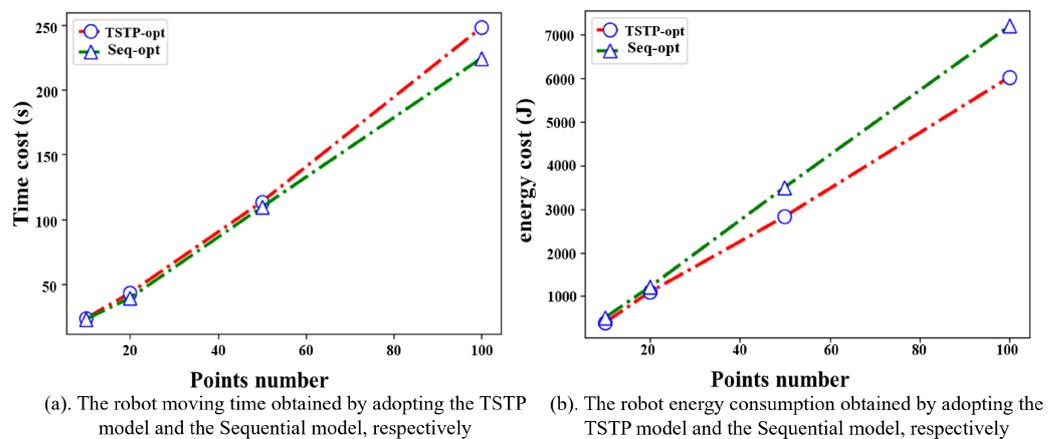


Figure 9. Comparison between the TSTP model and the sequential optimization model.

The relationship between a robot’s energy consumption and moving speed is complex and cannot be expressed by a simple linear model. The results of the experiments confirm that the TSTP model has the capability to effectively explore and strike a balance between

the moving speed of the robot and its energy consumption. This model is able to find solutions that minimize both the moving time of the robot and its energy consumption.

5.6. Comparison with Other Algorithms

To investigate the performance of the DQN, we conducted comparisons with swarm intelligence evolution algorithms. The comparison algorithms include GA (see [11]) and the differential evolution (DE) algorithm (see [18]), with a population size of 100, and other parameters match the reference. Each comparison is performed 10 times, and the average results are recorded.

Figure 10a shows the time–energy cost for all experiments. Figure 10b reports the computation time required by each algorithm. For small-scale problems, the algorithms find solutions of comparable quality because of the relatively small solution space and low computational complexity. As the problem scale increases, the proposed DQN algorithm outperforms the others. In particular, when the task points $m = 100$, the DQN algorithm can achieve an improvement of up to 28.80% compared to the GA algorithm. The DQN algorithm demonstrates a notably quicker decision process in all experiments, especially as the number of task points increases. For example, the computation time required by the proposed DQN algorithm is only a fifth of that of the DE algorithm when the number of task points $m = 100$. This is because the DQN algorithm employs a reinforcement learning framework, which focuses on learning from past experiences and gradually improves its performance. This framework enables the algorithm to make quicker and more informed decisions, resulting in reduced computational time. On the other hand, the DE algorithm typically requires a large number of iterations and evaluations of the objective function, resulting in longer computation times.

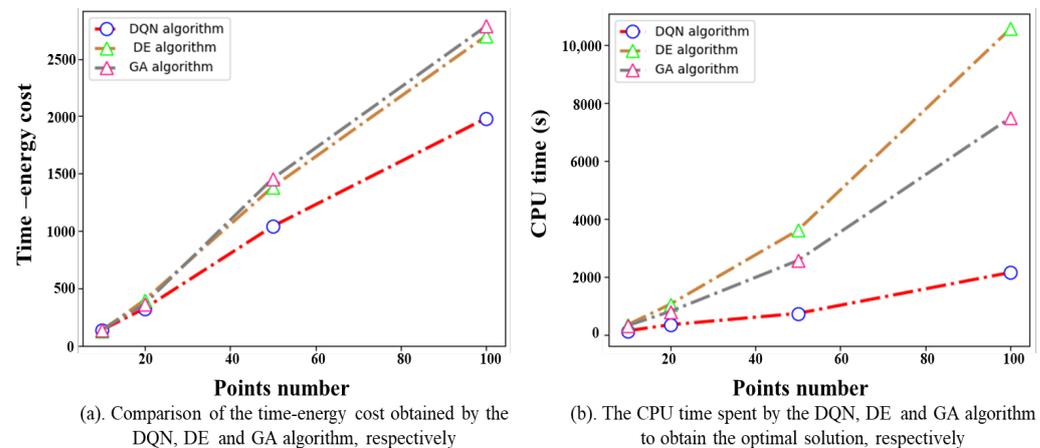


Figure 10. Performance comparison between the DQN algorithm and the others.

5.7. Physical Experiments

To validate the robustness and practicality of the proposed TSTP model and DQN approach, we carried out two sets of experiments on a real robot, where all parameter settings matched those of the previous simulation experiment.

5.7.1. The Validation of the Robot TSTP Model

In this section, two types of experiments are presented. The first one adopted the robot TSTP model, while the other one used the Seq-opt model; both took the DQN algorithm as the global optimization method and used the QPI algorithm for trajectory planning. To assess the effectiveness of the TSTP model across different complexities, tasks with 10 and 50 task points were included in the validation process; the experimental results are presented as follows.

The voltage and current values of the motor, along with their change curves, were documented during the robot's operation. Subsequently, the power curve and energy

consumption were calculated based on these data and are presented in Figure 11. When employing the TSTP model, the robot completed the task with 10 points in 12.179 s, resulting in an energy consumption of 1929.127 J. In contrast, when utilizing the Seq-opt model, the robot's movement took 14.303 s, consuming 2110.864 J of energy. Through analysis of the power curves, it is evident that the TSTP model guided the robot to execute tasks more steadily despite a slightly longer overall time, enabling efficient energy conservation. The superiority of the TSTP model was more obvious when the number of task points increased; see Figure 12. When using the TSTP model, the robot completed the task with 50 points in 42.86 s, resulting in energy consumption of 7212.83 J. In contrast, when utilizing the Seq-opt model, the robot's movement took 39.51 s, consuming 8819.45 J of energy. Despite the moving time being 8.47% longer than that of the Seq-opt model, the TSTP model resulted in an energy consumption that is 18.22% lower than the Seq-opt model.

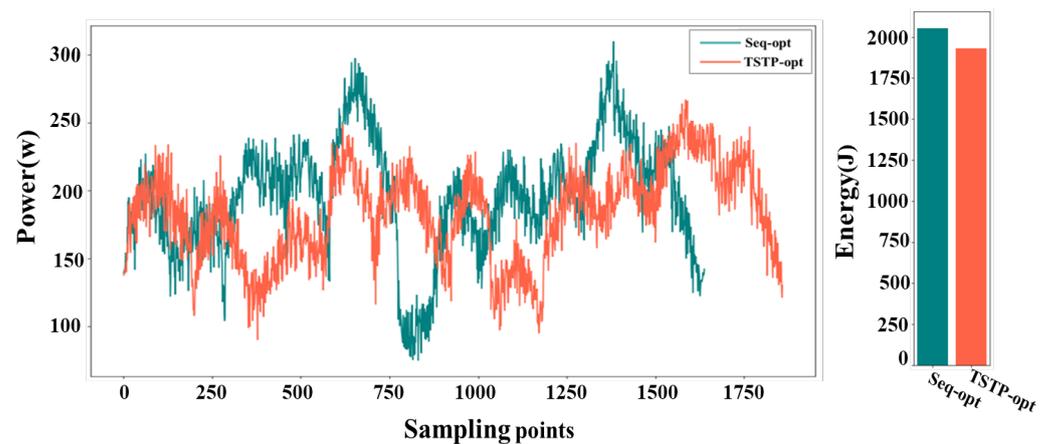


Figure 11. The power curves and energy consumption of the robot when executing the task involving 10 points using the TSTP model or Seq-opt model.

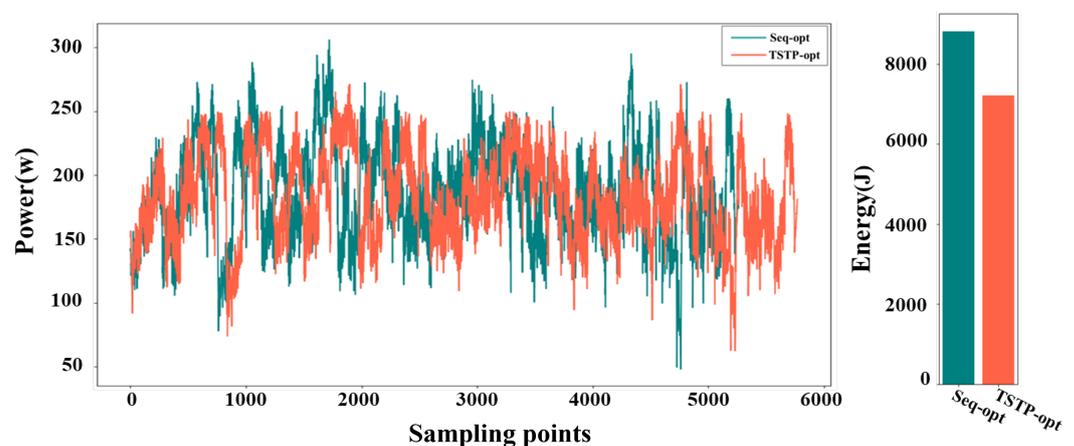


Figure 12. The power curves and energy consumption of the robot when executing the task of 50 points using the TSTP model or Seq-opt model.

5.7.2. The Performance Validation of the DQN Algorithm

In this section, we conducted three groups of experiments to test the performance of the DQN algorithm in solving the robot TSTP problem. The first group of experiments used the DQN algorithm, the second group used the DE algorithm, and the third group used the GA algorithm. Each group of experiments encompassed 10 and 50 task points. The TSTP model was employed in all experiments.

The voltage and current values of the motor, along with their change curves, were documented during the robot's operation. Subsequently, the power curve and energy consumption were calculated based on these data and are presented in Figure 13. Upon

analyzing the data, it is evident that the DQN algorithm is capable of proficiently addressing the robot TSTP problem. However, when the dataset was small, its performance was comparable to that of the traditional evolutionary algorithm. As the problem size increased, the superiority of the DQN algorithm became increasingly evident. For instance, in Figure 14, the optimal solution achieved by the DQN algorithm requires 17.83% less time and 25.28% less energy consumption compared to the DE algorithm, as well as 21.63% less time and 30.54% less energy consumption than the GA algorithm.

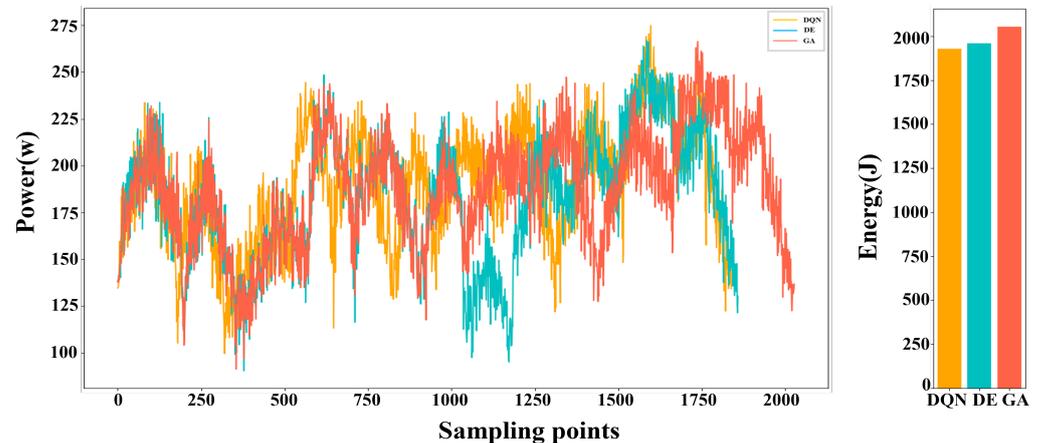


Figure 13. The power curves and energy consumption of the robot when executing the task of 10 points using the DQN algorithm, DE algorithm, and GA.

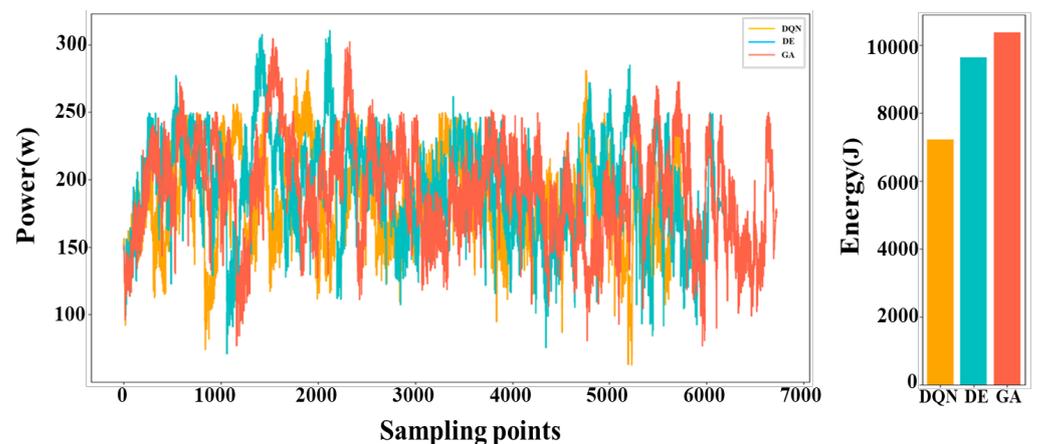


Figure 14. The power curves and energy consumption of the robot when executing the task of 50 points using the DQN algorithm, DE algorithm, and GA.

6. Conclusions

This work studied the simultaneous optimization of the robot task sequencing problem and trajectory planning problem, briefly called the robot TSTP problem. The objective is the simultaneous minimization of the robot's moving time and energy consumption. The robot TSTP problem is an NP-hard problem; it involves task point sequencing, IK solution calculation and selection, trajectory parameter determination, and trajectory planning. The solution space and computational complexity are much larger than those of any single optimization problem. To solve the robot TSTP problem, a hybrid method that combines the DQN algorithm and the QPI algorithm is proposed. First, the TSTP problem is modeled as a Markov decision process; then, we use the DQN algorithm to select the task points, IK solutions, and trajectory parameters at each decision step. Finally, the QPI algorithm is used to plan the moving trajectories between task points.

Through detailed experiments, we evaluated the competitiveness of our proposed DQN algorithm in solving the robot TSTP problem in terms of computation speed and

solution quality. In addition, the experimental results confirm the superiority of the TSTP model. The TSTP model extends the optimization space and enhances the DQN algorithm, enabling it to find more excellent solutions than any of the sequential optimization models. Furthermore, the TSTP model also increases the production flexibility of the robotic cell because it can easily adjust the scheduling and controlling parameters to respond to changes in production targets or conditions. Finally, when solving the TSTP problem, setting the time–energy optimization objective is more reasonable than setting a single objective, as it can explore a solution that can improve production efficiency while maximizing energy use. Currently, our research work only considers one robot’s production time–energy optimization, which limits the application of the proposed approach. The future direction is to apply these approaches to (a) multi-robot production environments, (b) stochastic dynamic production environments, and (c) a combination of the above.

Author Contributions: Methodology, X.D.; software, G.W.; investigation, P.Z.; data curation, S.C.; writing—original draft, X.D.; writing—review and editing, P.Z., C.S. and G.W.; project administration, P.Z.; funding acquisition, P.Z. and S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China [92267205, U1908212, and 92067205], the State Key Laboratory of Robotics of China (2023-Z15), the National Program for Funded Postdoctoral Researchers (GZB20230805).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare that they have no conflict of interest. The manuscript was written through the contributions of all authors. All authors have approved the final version of the manuscript.

References

- Chen, C.H.; Chou, F.I.; Chou, J.H. Optimization of robotic task sequencing problems by crowding evolutionary algorithms. *IEEE Trans. Syst. Man, Cybern. Syst.* **2022**, *52*, 6870–6885. [[CrossRef](#)]
- Zhou, J.; Cao, H.; Jiang, P.; Li, C.; Yi, H.; Liu, M. Energy-saving trajectory planning for robotic high-speed milling of sculptured surfaces. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 2278–2294. [[CrossRef](#)]
- Ratiu; Mariana; Prichici. Industrial robot trajectory optimization—A review. *MATEC Web Conf.* **2017**, *126*, 02005. [[CrossRef](#)]
- Alatartsev, S.; Stellmacher, S.; Ortmeier, F. Robotic task sequencing problem: A survey. *J. Intell. Robot. Syst. Theory Appl.* **2015**, *80*, 279–298. [[CrossRef](#)]
- Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [[CrossRef](#)]
- Zhao, L.; Fan, J.; Zhang, C.; Shen, W.; Zhuang, J. A drl-based reactive scheduling policy for flexible job shops with random job arrivals. *IEEE Trans. Autom. Sci. Eng.* **2023**, 1–12. [[CrossRef](#)]
- Kovács, A. Integrated task sequencing and path planning for robotic remote laser welding. *Int. J. Prod. Res.* **2015**, *54*, 1210–1224. [[CrossRef](#)]
- Kurtser, P.; Edan, Y. Planning the sequence of tasks for harvesting robots. *Robot. Auton. Syst.* **2020**, *131*, 103591. [[CrossRef](#)]
- Baizid, K.; Yousnadj, A.; Meddahi, A.; Chellali, R.; Iqbal, J. Time scheduling and optimization of industrial robotized tasks based on genetic algorithms. *Robot. Comput.-Integr. Manuf.* **2015**, *34*, 140–150. [[CrossRef](#)]
- FarzanehKaloorazi, M.; Bonev, I.A.; Birglen, L. Simultaneous task placement and sequence optimization in an inspection robotic cell. *Robotica* **2021**, *39*, 2110–2130. [[CrossRef](#)]
- Zacharia, P.; Aspragathos, N. Optimal robot task scheduling based on genetic algorithms. *Robot. Comput.-Integr. Manuf.* **2005**, *21*, 67–79. [[CrossRef](#)]
- Xidias, E.K.; Zacharia, P.T.; Aspragathos, N.A. Time-optimal task scheduling for articulated manipulators in environments cluttered with obstacles. *Robotica* **2009**, *28*, 427–440. [[CrossRef](#)]
- Zacharia, P.; Xidias, E.; Aspragathos, N. Task scheduling and motion planning for an industrial manipulator. *Robot. Comput.-Integr. Manuf.* **2013**, *29*, 449–462. [[CrossRef](#)]
- Gueta, L.B.; Chiba, R.; Ota, J.; Ueyama, T.; Arai, T. Coordinated motion control of a robot arm and a positioning table with arrangement of multiple goals. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 2252–2258.
- Wong, C.; Mineo, C.; Yang, E.; Yan, X.-T.; Gu, D. A novel clustering-based algorithm for solving spatially-constrained robotic task sequencing problems. *IEEE/ASME Trans. Mechatronics* **2020**, *26*, 2294–2305. [[CrossRef](#)]

16. Zahorán, L.; Kovács, A. Proseqqo: A generic solver for process planning and sequencing in industrial robotics. *Robot. Comput.-Integr. Manuf.* **2022**, *78*, 102387. [[CrossRef](#)]
17. Bobrow, J.E.; Dubowsky, S.; Gibson, J.S. Time-optimal control of robotic manipulators along specified paths. *Int. J. Robot. Res.* **1985**, *4*, 3–17. [[CrossRef](#)]
18. Lee, Y.D.; Lee, B.H.; Kim, H.G. An evolutionary approach for time optimal trajectory planning of a robotic manipulator. *Inf. Sci.* **1999**, *113*, 245–260. [[CrossRef](#)]
19. Saravanan, R.; Ramabalan, S.; Balamurugan, C. Evolutionary optimal trajectory planning for industrial robot with payload constraints. *Int. J. Adv. Manuf. Technol.* **2007**, *38*, 1213–1226. [[CrossRef](#)]
20. Liu, C.; Cao, G.-H.; Qu, Y.-Y.; Cheng, Y.-M. An improved pso algorithm for time-optimal trajectory planning of delta robot in intelligent packaging. *Int. J. Adv. Manuf. Technol.* **2019**, *107*, 1091–1099. [[CrossRef](#)]
21. Du, Y.; Chen, Y. Time optimal trajectory planning algorithm for robotic manipulator based on locally chaotic particle swarm optimization. *Chin. J. Electron.* **2022**, *311*, 906–914. [[CrossRef](#)]
22. Xu, Z.; Wang, W.; Chi, Y.; Li, K.; He, L. Optimal trajectory planning for manipulators with efficiency and smoothness constraint. *Electronics* **2023**, *12*, 2928. [[CrossRef](#)]
23. Zhao, J.; Zhu, X.; Song, T.; Meng, X. An improved whale optimization algorithm for robot time-jerk optimal trajectory planning. *J. Phys. Conf. Ser.* **2023**, *2170*, 012008. [[CrossRef](#)]
24. Rout, A.; Dileep, M.; Mohanta, G.B.; Deepak, B.; Biswal, B. Optimal time-jerk trajectory planning of 6 axis welding robot using tlbo method. *Procedia Comput. Sci.* **2018**, *133*, 537–544. [[CrossRef](#)]
25. Zhang, T.; Zhang, M.; Zou, Y. Time-optimal and smooth trajectory planning for robot manipulators. *Int. J. Control Autom. Syst.* **2015**, *19*, 521–531. [[CrossRef](#)]
26. Huang, Y.; Fei, M. Motion planning of robot manipulator based on improved nsga-ii. *Int. J. Control Autom. Syst.* **2018**, *16*, 1878–1886. [[CrossRef](#)]
27. Ma, J.; Gao, S.; Yan, H.; Lv, Q.; Hu, G. A new approach to time-optimal trajectory planning with torque and jerk limits for robot. *Robot. Auton. Syst.* **2021**, *140*, 103744. [[CrossRef](#)]
28. Li, X.; Zhao, H.; He, X.; Ding, H. A novel cartesian trajectory planning method by using triple nurbs curves for industrial robots. *Robot. Comput.-Integr. Manuf.* **2023**, *83*, 102576. [[CrossRef](#)]
29. Li, X.; Lan, Y.; Jiang, P.; Cao, H.; Zhou, J. An Efficient Computation for Energy Optimization of Robot Trajectory. *IEEE Trans. Ind. Electron.* **2022**, *69*, 11436–11446. [[CrossRef](#)]
30. Vergnano, A.; Thorstenson, C.; Lennartson, B.; Falkman, P.; Pellicciari, M.; Yuan, C.; Biller, S.; Leali, F. Embedding detailed robot energy optimization into high-level scheduling. In Proceedings of the 2010 IEEE International Conference on Automation Science and Engineering, Toronto, ON, Canada, 21–24 August 2010; pp. 386–392.
31. Meike, D.; Pellicciari, M.; Berselli, G. Energy Efficient Use of Multirobot Production Lines in the Automotive Industry: Detailed System Modeling and Optimization. *IEEE Trans. Autom. Sci. Eng.* **2014**, *11*, 798–809. [[CrossRef](#)]
32. Wigstrom, O.; Lennartson, B.; Vergnano, A.; Breitholtz, C. High-level scheduling of energy optimal trajectories. *IEEE Trans. Autom. Sci. Eng.* **2013**, *10*, 57–64. [[CrossRef](#)]
33. Hou, R.; Niu, J.; Guo, Y.; Ren, T.; Yu, X.; Han, B.; Ma, Q. A novel resolution scheme of time-energy optimal trajectory for precise acceleration controlled in dustrial robot using neural networks. *Actuators* **2022**, *11*, 130. [[CrossRef](#)]
34. Sang, W.; Sun, N.; Zhang, C.; Qiu, Z.; Fang, Y. Hybrid Time-Energy Optimal Trajectory Planning for Robot Manipulators with Path and Uniform Velocity Constraints. In Proceedings of the 2022 13th Asian Control Conference (ASCC), Jeju, Republic of Korea, 4–7 May 2022; pp. 334–340.
35. Shi, X.; Fang, H.; Pi, G.; Xu, X.; Chen, H. Time-Energy-Jerk Dynamic Optimal Trajectory Planning for Manipulators Based on Quintic NURBS. In Proceedings of the 2018 3rd International Conference on Robotics and Automation Engineering (ICRAE), Guangzhou, China, 17–19 November 2018; pp. 44–49.
36. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [[CrossRef](#)]
37. Xidias, E.; Moulianitis, V.; Azariadis, P. Optimal robot task scheduling based on adaptive neuro-fuzzy system and genetic algorithms. *Int. J. Adv. Manuf. Technol.* **2021**, *115*, 927–939. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.