IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 February 2023, accepted 23 March 2023, date of publication 27 March 2023, date of current version 7 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3262169

RESEARCH ARTICLE

Tool Path Optimization for Complex Cavity Milling Based on Reinforcement Learning Approach

YI WAN[1], (Member, IEEE), WEI XU[2], AND TIAN-YU ZUO[3]
[1]School of Environmental Science, Nanjing Xiaozhuang University, Nanjing 211171, China
[2]School of Mechanical and Electrical Engineering, Sanjiang University, Nanjing 210012, China
[3]School of Automation, Nanjing University of Information Science and Technology, Nanjing 210044, China

Corresponding author: Yi Wan (wan_sju@163.com)

This work was supported in part by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant 19KJB460006, and in part by the China Postdoctoral Science Foundation under Grant 2019M651642.

ABSTRACT In the machining of parts, tool paths for complex cavity milling often have different generation options, as opposed to simple machining features. The different tool path generation options influence the machining time and cost of the part during the machining process. Decision makers prefer tool path solutions that have fewer blanking lengths, which means that the machining process is more efficient. Therefore, in order to reduce costs and increase efficiency, it is necessary to carefully design the tool path generation for the features to be machined on the part, especially for complex cavity milling features. However, solutions to the problem of optimal design of tool paths for complex cavity milling features have not been well developed in current research work. In this paper, we present a systematic solution for complex cavity milling tool path generation based on reinforcement learning. First, a grid converter is executed for converting the 3D geometry of the cavity milling feature into a matrix of planar grid points recognisable by the program, set according to the cutting parameters. Afterwards, the tool path generation process is refined and modelled as a Markov decision process. Ultimately, a tool path generation solution combining the A* algorithm with the Q-learning algorithm is executed. The agent iterates through trial and error to construct an optimal tool path for a given cavity milling task. Three case experiments demonstrate the feasibility of the proposed approach. The superiority of the reinforcement learning-based approach in terms of solution speed and solution quality is further demonstrated by comparing the proposed approach with the evolutionary computational techniques currently popular in research for solving tool path optimisation design problems.

INDEX TERMS Tool path optimization, cavity milling, path planning algorithm, reinforcement learning, Q-learning algorithm.

I. INTRODUCTION

Numerical control (NC) technology and the resulting Computerised Numerical Control (CNC) machine tool have been used as the basis for the industrial revolution in manufacturing. 2D milling is the most common type of CNC milling used in normal part machining processes [1], [2]. This is due to the fact that most of the mechanical parts that are widely demanded in practical engineering are 2.5D, and even complex shape-machined parts such as surface engraving, stamping moulds and propellers can be manufactured by rough machining of the blank 2.5D with 3D-5D finishing for the entire manufacturing process [3], [4].

In 2D cavity milling tasks (*e.g.* cavity milling with islands), the tool path generation scheme has a serious impact on machining efficiency. For example, common tool path generation approach include zigzag cutting mode, offset loop cutting mode and composite mode including zigzag and offset

The associate editor coordinating the review of this manuscript and approving it for publication was Abderrahmane Lakas.

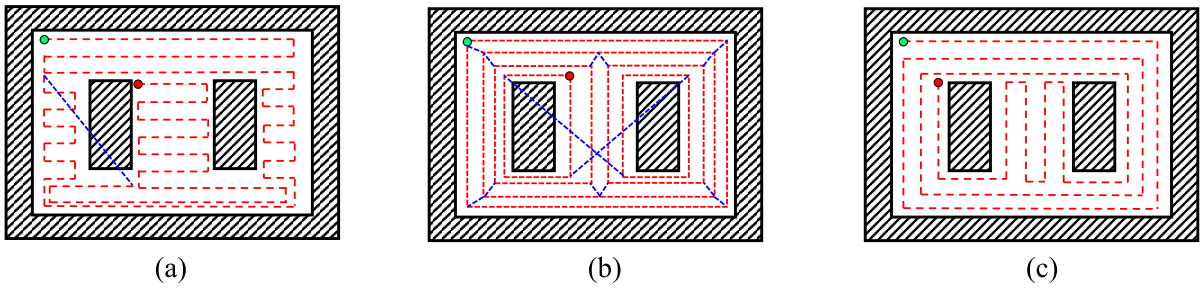

FIGURE 1. Three processing tool path schemes for a cavity milling feature with two isolated islands. (a) path generation scheme based on zigzag cutting mode; (b) path generation scheme based on offset loop cutting mode; (c) optimal path generation scheme. The green and red dots in the diagram indicate the processing start point and the processing end point respectively. The red dotted line shows the actual cutting path of the tool. The blue dotted line shows the non-cutting path of the tool.

loop. If these approach are used directly, the entire tool path generated is discontinuous and there are a large number of void cutting. As shown in Fig. 1(a) and (b), for a 2D cavity milling feature with two isolated islands, the conventional tool path generation scheme produces a haphazard path with a large number of redundant cutting throughout the path. The tool path shown in Fig. 1(c), on the other hand, is more organised and has fewer redundant cutting. It can be said that the tool path generation solution shown in Fig. 1(c) is preferred by decision makers when faced with complex cavity milling tasks. It is therefore necessary to carefully tailor the tool paths to the characteristics of the part to be machined during the actual manufacturing process. This requirement is abstracted into an optimisation problem, the tool path optimisation problem, called TPO for short.

The level of interest in TPO research is currently growing. For different types of machining features, this paper broadly classifies TPO into TPO based on flat milling and TPO based on surface milling, *i.e.,* 2D-TPO and 3D-TPO.

For 3D-TPO, Zhou et al. [5] considered the energy consumption and environmental impact of the machining process and developed a multi-objective tool path optimisation model with maximum machining efficiency, minimum energy consumption and minimum carbon emission. Xu et al. [6] provided a simple tool path generation solution for machining complex shaped part surfaces on a five-axis machine. The proposed approach was experimented in both computer simulations and physical cutting. Zhao et al. [7] developed an analytical decoupled angle smoothing approach for the tool path smoothing and generation problem in real time for five-axis machines.

As can be seen, most of the extant research has focused on the 3D-TPO problem, while only a small amount of work has explored 2D-TPO in a preliminary way. However, since 2D cavity milling features are not only the most commonly encountered type of machining feature but also form the basis for complex machining tasks, there is an urgent need to develop a systematic solution for 2D-TPO. The main focus of this paper is on 2D-TPO in the presence of complex cavity features. Barclay et al. [8], for example, used genetic programming techniques to develop a generic program which automatically generates the optimal tool path based on the contour shape of a given 2D milling area. Zhou et al. [5] developed a genetic algorithm (GA) based tool path generation scheme specifically for complex cavity milling tasks. In their approach, the 3D milling region is projected onto a 2D layer consisting of a grid. Afterwards, TPO is transformed into a problem of optimising the node access order of the tool on the 2D layer, which is a combinatorial optimisation problem and an NP-hard problem. The GA-based tool path generation scheme has been shown to be significantly superior compared to common tool path generation schemes [9], [10], [11], [12], [13]. However, evolutionary computation is often accompanied by the drawback of being prone to local optimal solutions. As the size of the problem increases, its iterative operations are not efficient. More studies related to tool paths also exist [14], [15], [16], [17], [18], [19], [20], [21].

We want to highlight the practical value of developing TPO from an application perspective. This is also the starting point of this work. *TPO is a key part of the smart green manufacturing system [22], [23], [24], and it is often used as a sub-problem in the process of optimising the manufacturing system for energy saving and emission reduction.* Solving TPO efficiently and with high quality is very beneficial to the solution of the upper-level problem. For example, in [5], the authors construct a two-layer optimisation model for optimising the carbon emissions of a part machining process. In their proposed optimisation approach, both the cutting parameter optimisation problem and the TPO are solved by means of evolutionary computation-based approach. The problem structure of this model and the optimization-evaluation process are shown in Fig. 2. It is can be seen that the efficiency of the TPO as a sub-problem has a significant impact on the efficiency of the whole two-layer optimisation model [24], [25], [26], [27], [28]. Clearly, the use of an iterative-based optimisation approach to TPO is inappropriate in manufacturing system optimisation problems. As an alternative heuristic optimisation paradigm, the feasibility and effectiveness of applying reinforcement learning (RL) techniques to TPO has not been explored. It is expected to solve TPO efficiently and with high quality. Inspired by this, in this paper we develop

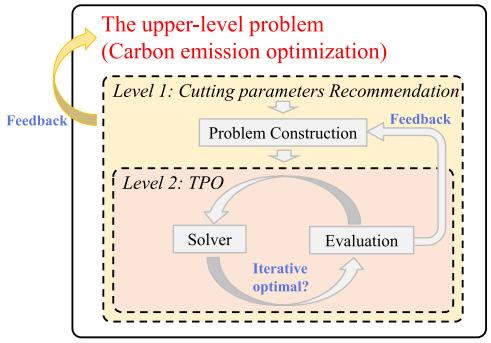

FIGURE 2. The problem structure and optimization process of the multilevel optimization model with TPO as a subproblem.

an RL-based tool path generation scheme for complex cavity milling tasks. The proposed approach consists of three components, a grid converter, RL modelling of TPO and a tool path generator combining the A* algorithm [29], [30] with the Q-learning algorithm [31], [32], [33]. Firstly, a grid converter is used to map the 3D cavity profile structure into a grid plane layer based on the given cutting parameters. In this way, the solution of TPO can be transformed into a problem of optimising the node access order of the tool in the grid plane. The node-access order optimisation problem is then modelled as a Markov Decision Process (MDP) [34], [35]. The states, actions and rewards of the tool in the grid plane are then fully defined. Finally, a tool path generator combining the A* algorithm with the Q-learning algorithm is executed to automatically output the optimal tool path for a given cavity machining profile. Several case studies demonstrate the feasibility of the RL-based solution for solving TPO. The superiority of the RL-based solution is also demonstrated by comparing it with an evolutionary computation-based approach.

The main contributions of this paper are as follows.

(1) To the best of our knowledge, this work is the first to formulate the TPO problem as a Markov decision process. The correspondence between the states, actions and rewards and the tool path factors in the machining process is constructed.

(2) An RL-based tool path generation framework is proposed. Its biggest advantage is the ability to produce high-quality tool paths in a short time (*i.e.,* less redundant cutting travel and void cutting).

(3) Our work is also the first to highlight and reveal the high requirements of TPO optimizers in terms of solution speed. Extensive numerical experiments show that our proposed method is not only superior to the comparison methods in terms of solution quality, but also can achieve excellent solutions in a very short running time. This advantage makes our method more suitable for solving multi-stage manufacturing system optimization problems with TPO as a subproblem.

The rest of this paper is organised as follows. Section II presents the TPO model based on a Markovian Decision Process. The details of the proposed method are elaborated in Section III. In Section IV, several case studies and comparative experiments are executed. Section V discusses the limitations of our method. Section VI summarizes the paper and points out future research directions.

II. PROBLEM FORMULATION

A. MDP AND BELLMAN OPTIMIZATION EQUATION

1) MDP

MDPs are sequential decision problems for fully perceptible stochastic environments with Markov transfer models and additional rewards. Typically, MDP is used to construct the state and environment in RL algorithms. In other words, the agent-environment interaction process can be assumed to be an MDP. an MDP can be represented using a quadruplet, namely $\langle S, A, P, R\rangle$. where S denotes a finite set of states that covers all possible states in the environment and that the agent may face. A denotes a finite set of actions, which covers the set of action types that the agent can take. It is important to note that the set of executable actions of an agent may vary across states, but must all be subsets of A. P denotes the state transfer probability matrix. When the agent picks an action a_t in state s_t, the environment will determine the successor state s_{t+1} based on P. It should be noted that for the same state-action sequence (s_t, a_t), the subsequent states s_{t+1} can be fixed or not, depending on the properties of the problem. R denotes the reward function. When the agent completes a state transfer the environment will feed the agent a reward value based on (s_t, a_t, s_{t+1}). Typically, a discount parameter $\gamma \in [0, 1]$ needs to be introduced to weigh the relative importance of future and timely rewards. When the agent is at step t, the reward with discount factor is defined as follows.

$$R_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_k = r_t + \gamma R_{t+1}, \tag{1}$$

where r_k is the value of the prompt reward fed by the environment at step k. It can be seen that the above equation is an iterative equation over R_t with R_{t+1}.

2) BELLMAN OPTIMALITY EQUATION

The agent will start in an initial state s_0 and continuously interact with the environment through action selection and state transfer, and the environment will provide the agent with timely rewards based on the agent's behaviour. The agent learns a policy π by trial and error, which can be used to guide the agent's action choices in any state. Value functions are used to evaluate whether a policy is a better policy or not. The value function is divided into a state-value function $v_\pi(s)$ and a state-action value function $q_\pi(s, a)$. They are the expected values used to evaluate the sum of the future rewards that an agent will receive given the current state and the state-action binary, respectively [36]. In general, for a given arbitrary state $s \in S$, the larger the value of the value function, the superior

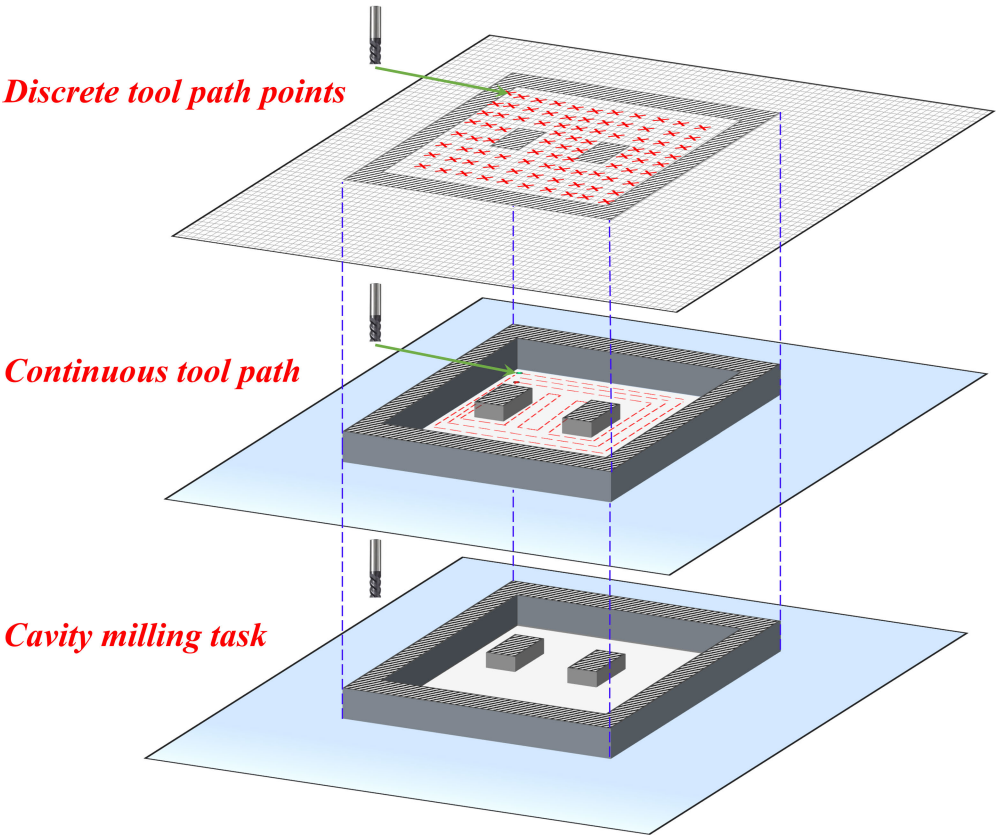

FIGURE 3. Converts continuous tool path of a 3D cavity milling model into discrete tool path points on a 2D plane. (a) a given cavity milling task; (b) a tool path scheme (continuous path) for the given milling task; (c) a plane formed by the discrete tool path points corresponding to the given path scheme.

the policy.

$$v_\pi(s) = \mathbb{E}_\pi\left[r_{t+1} + \gamma R_{t+1} \middle| S_t = s\right]. \quad (2)$$

$$q_\pi(s,a) = \mathbb{E}_\pi\left[r_{t+1} + \gamma R_{t+1} \middle| S_t = s, A_t = a\right]. \quad (3)$$

The ultimate goal of the RL algorithm is to construct an optimal policy π^*. Therefore, all carefully designed algorithms are designed to efficiently solve for the policy π^* corresponding to the optimal state-value function v^* and the optimal state-action value function q^*, a process that corresponds to the Bellman optimality equation, as follows.

$$v^*(s) = \max_a \sum_{s',r} p\left(s', r \middle| s, a\right)\left[r + \gamma v^*\left(s'\right)\right]. \quad (4)$$

$$q^*(s) = \sum_{s',r} p\left(s', r \middle| s, a\right)\left[r + \gamma \max_{a'} q^*\left(s', a'\right)\right]. \quad (5)$$

There are many extant algorithms for solving optimal policy π^* according to the Bellman optimisation equation. They can be roughly divided into value-based schemes and policy-based schemes. The value-based schemes are easier to implement than the policy-based schemes. One representative of a value-based scheme is Q-learning. It is an iterative optimisation scheme based on the Time Difference (TD) algorithm [37] and the Bellman Optimisation Equation [33]. In this paper, Q-learning is used to develop tool path generation approach. The specific deployment of Q-learning in TPO is described in detail in Section III-C.

B. AGENT AND ENVIRONMENT

The tool used to machine a given cavity feature is considered as an agent. In each interlude, the agent continuously interacts with the environment. The area to be machined for the given cavity feature is considered as the environment. It is important to note that different agents (or different tools) have an impact on the environment. This is because the tools have different parameters, such as the diameter of the milling cutter. In the machining of cavities, those with larger diameters can cut a large amount of metal in a single step, so that in this case fewer tool travel steps are required to complete a given cavity milling task than with smaller diameter milling tools.

C. STATE

As Fig. 3 shows, the state of the agent in the environment will be well defined if we transform the continuous tool path into discrete path points. When the tool moves from path point a to b, it is geometrically equivalent to the metal that has been removed from between a and b by the tool.

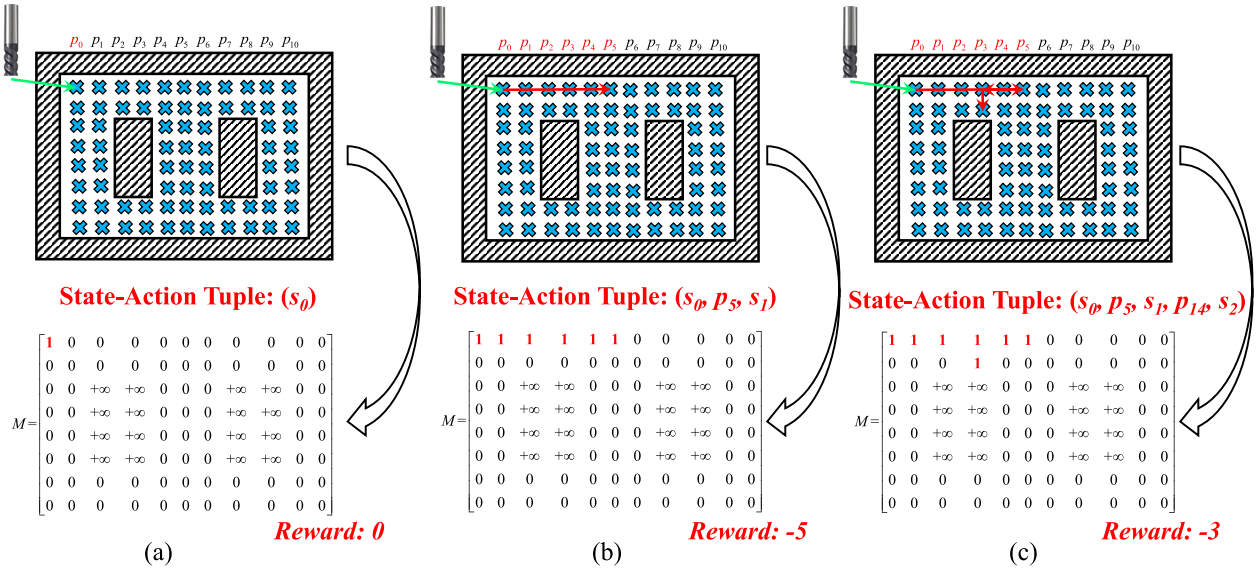

FIGURE 4. Correspondence between the environment and the feature matrix M. (a) initial state s_0, tool departs from p_0; (b) tool selects action p_5 in previous state and moves to new state s_1, reward value of environment feedback is -5; (c) tool selects action p_{14} in s_1 and moves to new state s_2, reward value of environment feedback is -3, where $L_a = 1$ and $L_v = 2$. The values of the elements in the feature matrix are continuously updated with the environment.

Consider that there is only one element in the environment, the discrete path point. Therefore, a feature matrix can be created to represent the current state of the environment. This is shown in Fig. 4, where 1 indicates a point that has been cut, 0 indicates a point that has not been cut and $+\infty$ indicates the outer contour of the cavity, which is the area that cannot be cut.

D. ACTION AND REWARD

Suppose that $P = \{p_0, p_1, p_2, \ldots, p_{n-1}\}$ is the set of all n path points. At the initial moment, the agent starts from a fixed starting point p_0, as shown in Fig. 4(a), after which p_0 is immediately removed from P and the next visited path point p_k is selected. Then, the tool starts walking from p_0 and ends up at p_k and all path points visited along the path are set to 1. If the state of a point is already 1, it remains 1. Fig. 4(b) gives an example. The tool starts from p_0 and selects p_5 as the target point. It can be noticed that the agent completes the cut from p_1, p_2, p_3, p_4 and p_5, even though it has only selected one step of the action.

It is important to note that since points that have been visited will be dynamically and promptly removed from the set of path points P. Therefore, for any moment t, any node that belongs to P is available for selection. Two cases arise in this case: (1) actual cutting and (2) void cutting. An example of actual cutting is shown in Fig. 4(b), where the tool starts from p_0, moves horizontally to p_5 and ends. The indirectly accessed path points p_1, p_2, p_3 and p_4 during this movement are path points that have not been accessed before the tool performs the given action (*i.e.,* from p_0 to p_5). Fig. 4(c) gives an example of an void cut when the tool has to pass through path points that have already been visited. In this case the tool starts at p_5 and selects a target point of p_{14}. In order to complete this movement, the tool may need to return from p_5 to p_4 and then to p_{14}, depending on the travel rules specified in advance. The return from p_5 to p_4 is an void cut, since the metal between p_4 and p_5 has already been removed in a previous cutting action.

In this paper, the optimisation objective of TPO is to minimise the path length of the actual cut as well as the path length of the void cut. The former is to improve machining efficiency and the latter to reduce unnecessary energy consumption. The rewards obtained by the agent during its interaction with the environment, fed back by the environment, are defined as follows.

$$r\left(a_t \mid s_t, M\right) = -\left(L_a + L_v\right) = -\sum_{p_i, p_j \in P, i \neq j} x_{i,j} \cdot \overline{p_i p_j}. \quad (6)$$

where $r\left(a_t \mid s_t, M_t\right)$ denotes the reward value obtained by choosing a_t progeny heeded in state s_t, which is defined as the negative of the sum of the actual cutting path length (*i.e., L_a*) as well as the void cutting path length (*i.e., L_v*). $x_{i,j}$ is a decision variable. It represents the number of links that exist between points i and j at state s_t. If no link exists between i and j, it is 0, which means that the metal between i and j has not been cut yet. Conversely, the value is equal to the number of times the tool has cut between i and j (both actual and void cuts). $\overline{p_i p_j}$ denotes the distance between two points, defined in general as the Euclidean distance. Based on the above definition we can obtain a reward of -5 for the action shown in Fig. 4(b) and a reward of -3 (*i.e.,* $-1\text{-}2 = -3$) for the action shown in Fig. 4(c).

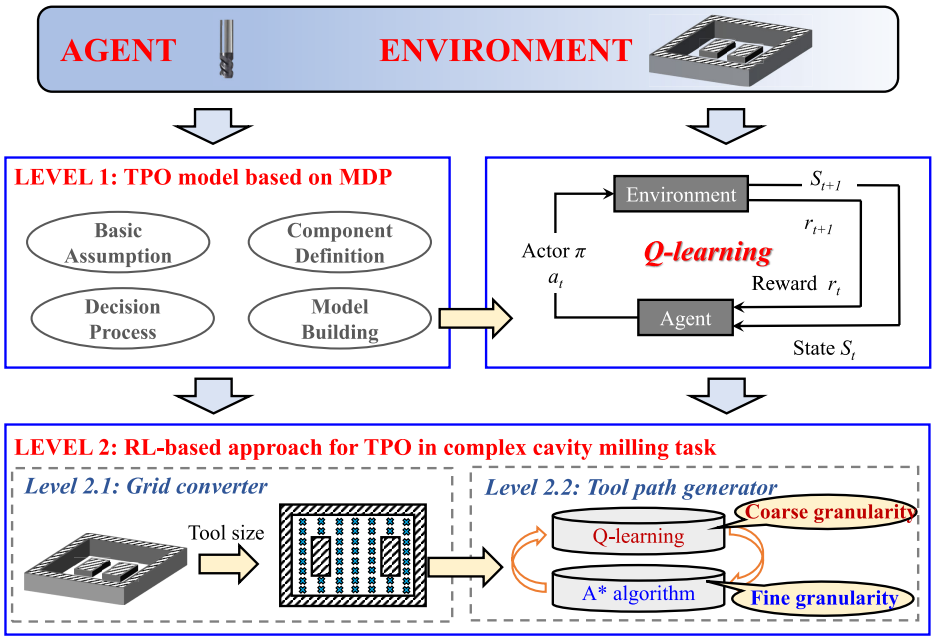

FIGURE 5. The procedure framework for the proposed approach.

E. STATE PROBABILITY TRANSITION AND END TIME

The state transfer probability of TPO's MDP model depends on the travel rule of the tool. When the tool travel rule between any two points is deterministic, the state transfer probability during cavity milling is determined. Conversely, if the travel rule is non-deterministic (*e.g.* a heuristic rule), the state transfer probability is non-deterministic. In this paper, the A* algorithm is used as a way to determine the specific walking process of the tool between any two points. It is a heuristic rule that is often applied in path planning problems. The specific deployment of the A* algorithm is described in detail in Section III-C. Thus, when the A* algorithm is applied to fine-grained tool path planning, the state transfer probability matrix of the MDP model corresponding to TPO is unknown. This inspired us to use a model-free reinforcement learning algorithm to solve coarse-grained tool path planning. It also justifies the deployment of Q-learning for solving TPO and the necessity of combining the A* algorithm with Q-learning.

An episode is considered to be over when all element values in the feature matrix M are greater than or equal to 1, which means that the given cavity milling task is complete. The agent continuously interacts with the environment one interlude after another, and when one interlude ends, a completely new one starts immediately. This is done until the maximum number of iterations is reached. Through trial and error and experience (*i.e.,* Q table) an optimised tool path is finally output.

III. PROPOSED APPROACH

A. OVERVIEW

The framework of the RL-based tool path generator proposed in this paper is shown in Fig. 5. It contains two levels of tasks. Level 1 is aimed at modelling, using MDP theory to build TPO models that can be solved by the RL algorithm. Level 2 is the approachology, which consists of a grid converter and a tool path generator combining the A* algorithm with the Q-learning algorithm. The A* algorithm is used for fine-grained path planning and the Q-learning algorithm is used for coarse-grained path planning. Intuitively speaking, the Q-learning algorithm is used to develop an actor for macroscopic selection of actions (*i.e.,* target path points) in a continuous process of interoperability with the environment, while the A* algorithm is considered to be the executor. Which is responsible for concretely completing the action instructions output by the actor, *i.e.,* feeding back the tool walk from the current path point to the target path point. The modelling process for level 1 has been well described in Section II. The deployment of the modules of the proposed approachology for solving TPO, including the grid converter and the tool path generator, are described in Sections II and III respectively.

B. GRID CONVERTER

The role of the grid converter is to map the TPO problem of a 3D part to be milled into a 2D planar problem of optimising the sequence of the tool path points. For a given cavity milling task, it is first necessary to distinguish between the area to be cut and the area that cannot be cut. The contour of the cavity is determined taking into account the allowable machining error Δ and the tool radius R. The contour of the cavity is divided into an outer and an inner contour. The area enclosed between them is the area to be cut, other than this the area on the part is the non-cuttable area. Afterwards, an empty grid area Φ is constructed based on the cutting width (*i.e.,* tool diameter), with each sub-square grid having a length

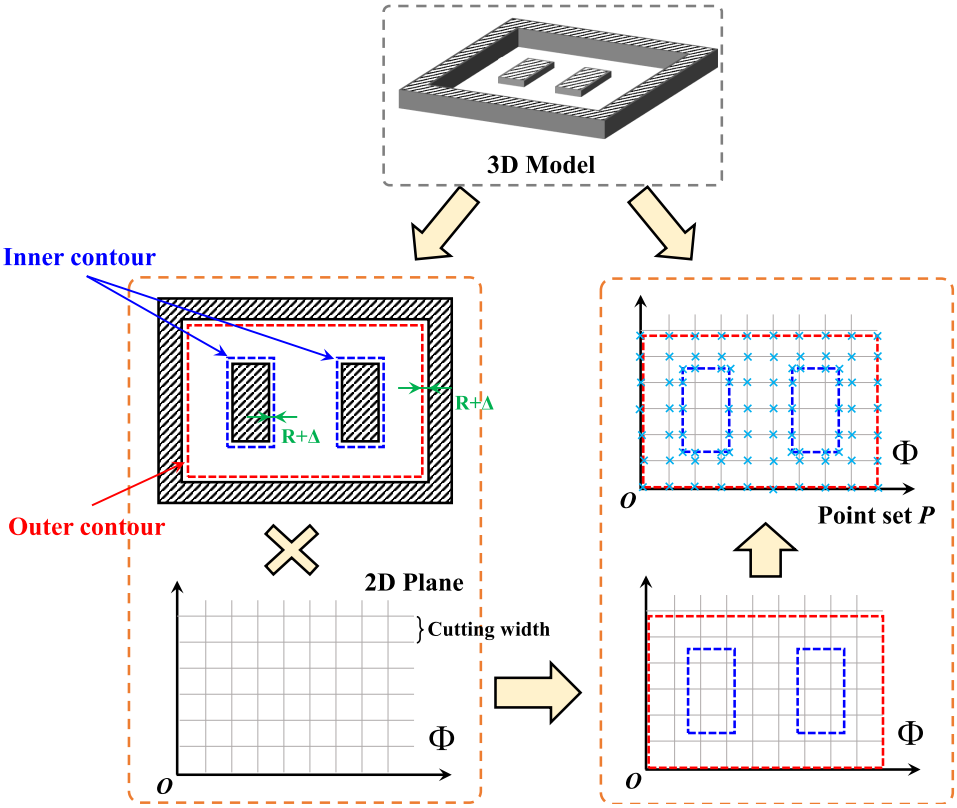

FIGURE 6. The detailed steps of the proposed grid converter.

equal to the cutting width. Next, one of the vertices of the contour line of the cavity is placed at the origin of Φ and it is ensured that Φ covers the contour line. Finally, the grid intersections in Φ surrounded by contour lines (inside the outer contour line and outside the inner contour line, to be precise) and the intersections of the contour lines with the grid lines are identified and combined as a set of tool path points. Fig. 6 illustrates the execution flow of the grid converter. It should be noted that the continuous tool path is not broken by the definition of the machining area as a discrete set of points. A continuous path can be expressed by directionally connecting multiple discrete path points. As shown in Eq. 6, the length of a continuous path is equal to the sum of the distances of the connected discrete points.

C. TOOL PATH GENERATOR

1) COARSE GRAINED PATH PLANNING: Q-LEARNING

Following the definition given in Section II, the size of the state set S and the size of the action set A are both equal to the size of the discrete point set P. In normal machining, the size of the point set P of the discretized cavity is usually not very large. There is no need to go to the trouble of using deep reinforcement learning (DRL) approaches such as deep Q-network [38], [39] or actor-critic approach [40], [41] to solve TPO problem. Instead, it is more appropriate to use the comparatively lighter Q-learning approach to solve TPO.

Q-learning is deployed with the aim of training an actor to progressively control the tool movement through continuous iterative trial and error. Once trained to perfection, the actor can output an optimised tool path by virtue of the q value in the Q-table and according to the ϵ-greedy algorithm.The specific procedure used by Q-learning to solve TPO is shown in Alg. 1. First, the Q-table Q will be initialised and it will be of size $P \times P$ (line 1). This is followed by an iterative trial-and-error phase. Before each new episode is started the feature matrix M is reloaded, the action set A, the state *state* is initialised, the cumulative reward value *Reward* for the current episode is initialised and the tool path recorder *Path* for the current episode is initialised (lines 3 to 7). Then, as long as there are still 0 elements in M, which means that the milling task for the current episode has not yet been completed, the action a is continuously selected from the action set A according to the ϵ-greedy algorithm, along with the current state *state* (line 9). Once the agent has selected an action a, the environment is immediately updated. Specifically, the agent simulates the execution of a in the current state *state* and records all tool path points passed along the way during the execution of a until the target point (*i.e., a*) is reached and then gives back the latest state *state*$'$, the reward value of the action r and the updated feature matrix M used to represent the environment (line 10). Then, the reward value r for the action, the action a and all tool path points passed

Algorithm 1 The Procedure of the Q-Learning Approach for TPO Problem

Input: The set of discrete points for a given cavity milling task P; Greedy factor ϵ; Discount factor γ; Learning rate lr.

Output: The optimal tool path *Path*.

1: Initialize Q table Q;
2: **while** terminal condition **is not** met **do**
3: Initialize M;
4: $A \leftarrow P$;
5: $state \leftarrow 0$;
6: $Reward \leftarrow 0$;
7: $Path \leftarrow \{\}$;
8: **while** $all(M)$ **is** *False* **do**
9: $a \leftarrow$ TakeAction($state, A$); // according to ϵ-greedy approach.
10: $state', r, M \leftarrow$ StepEnv($state, a$); // call A* algorithm for executing the given action a.
11: $Reward \leftarrow Reward + r$;
12: $Path \leftarrow \{Path, points\}$; // *points* is the point set containing the a and the points which along the way to execute a.
13: Remove a and the points which along the way to execute a from A;
14: $error \leftarrow r + \gamma * (max(Q[state', :]) - Q[state, a])$; // TD error.
15: $Q[state, a] \leftarrow Q[state, a] + lr * error$;
16: $state \leftarrow state'$;
17: **end while**
18: **if** *Reward* **is** the best **then**
19: Save *Path*;
20: **end if**
21: **end while**
22: **return** *Path*

along the way when executing a are recorded in *Reward* and *Path* respectively (lines 11 and 12). All points that have already been selected or passed along the path during the execution of a given action are removed from A and cannot be used as candidates for the next action selection (line 13). Next, the Q table is updated according to the TD algorithm (lines 14 and 15) and the current state *state* is transferred to the latest state $state'$ (line 16). At the end of the current episode, the cumulative reward value *Reward* of the current episode is judged to be the best in history, the smaller the value the better, and if it is the best, then *Path* is saved and the next episode (lines 18 to 20) is proceeded directly. This process is repeated until the termination condition is met (line 2). Eventually a historically optimal *Path* will be output by Alg. 1 (line 22).

2) FINE GRAINED PATH PLANNING: A* ALGORITHM

Actors learned by the Q-learning algorithm are used to guide the agent through a given cavity milling task at the macro level. In a given state, the actor selects an action a from the action candidate set A based on the empirical values in the Q table. Afterwards, the A* algorithm acts as an actor giving the process of moving from the current starting point (*i.e.,* state move) to the target point (*i.e., a*). It can be said that the role of the A* algorithm is to take care of fine-grained tool path planning. This means that given the position of the starting point and the target point, the A* algorithm will output a valid path connecting these two points.

However, in general, the A* algorithm does not guarantee the shortest path between the two points, but rather balances solution efficiency with solution quality, improving solution efficiency while minimising the loss of solution quality. However, the A* algorithm is always able to plan satisfactory results in simple cases, which is why it is so popular in path planning problems. In TPO the problem of optimising the milling path of a tool is transformed into an optimisation problem of the order of access to the discrete points of the tool on the plane. The set of discrete points is uniformly distributed on the grid plane, constructing a natural map scene for the A* algorithm, and the map area is uncomplicated and small in size. It is therefore worthwhile to trust that the A* algorithm can effectively act as an actor to concretely implement the action commands issued by the actor, return satisfactory action execution results, and ultimately use these results as a basis for the agent to feed back updated environmental states as well as reward values (including actual cut lengths as well as meaningless void tool lengths).

It is important to note that although the A* algorithm is a heuristic, its output is somewhat randomised. However, in simple map scenarios, as in type-cavity maps, the inherent randomness of the A* algorithm is not sufficient to significantly interfere with the actor's action selection. We will demonstrate this in numerical experiments.

IV. NUMERICAL EXPERIMENT

In this section, we will optimize the cavity milling tool paths for four case workpieces to demonstrate the effectiveness of the algorithms proposed in this paper and their superiority over the comparison approach. All algorithms are coded in Python 3.8, and experiments are performed on an Intel Core i5-7300HQ (2.5GHz) with 8 GB RAM.

A. TEST CASE

The 3D models of the four cavity milling cases used in this experiment are shown in Fig. 7. The content of the main chemical components and the metal types of the four test parts are shown in Table 1. They are all metals that are commonly used in real-life scenarios. The constructed cavities are also derived from the actual machining requirements of the manufacturing company. Workpiece 1, for example, is a milling task with two islands, which is a frequently encountered feature in normal machining tasks and is often used as a classic test case in tool path optimisation studies.The milling cutter brand used for machining is EMC54120 4F and its detailed parameters are shown in Table 2.

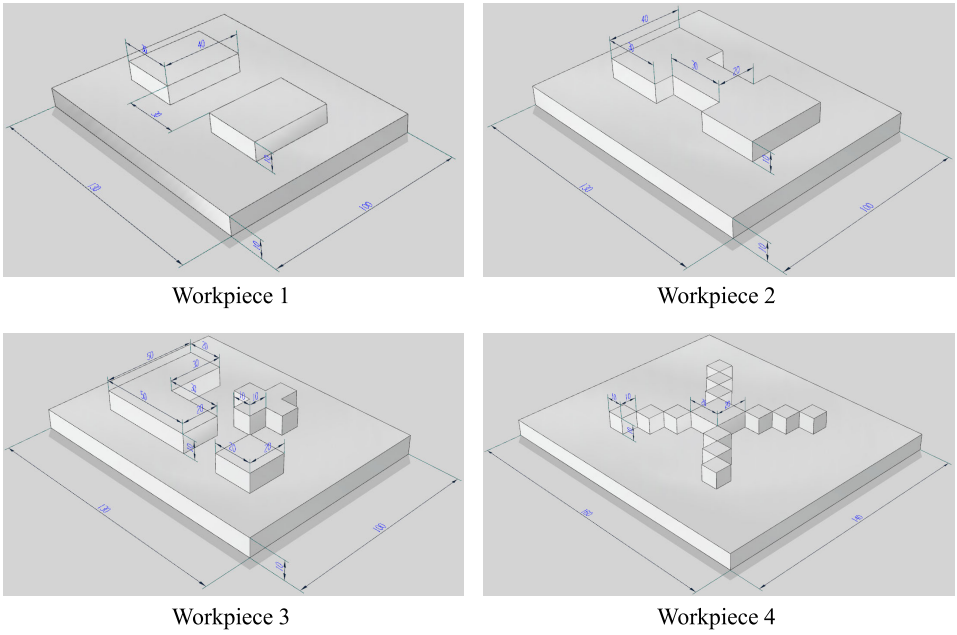

FIGURE 7. The 3D model with PMI dimensioning of the four test workpieces used in the experiment.

TABLE 1. Chemical composition of the four test workpieces (%).

	Carbon	Chromium	Manganese	Nickel	Phosphorus	Sulfur	Silicon	Brand
Workpiece 1	0.42-0.50	⩽0.25	0.50-0.80	⩽0.25	⩽0.035	⩽0.035	0.17-0.37	# 45
Workpiece 2	0.18-0.24	0.70-1.00	0.50-0.80	⩽0.30	⩽0.035	⩽0.035	0.17-0.37	20Cr
Workpiece 3	0.37-0.44	0.80-1.10	0.50-0.80	⩽0.30	⩽0.035	⩽0.035	0.17-0.37	40Cr
Workpiece 4	0.42-0.50	⩽0.25	0.50-0.80	⩽0.25	⩽0.035	⩽0.035	0.17-0.37	# 45

TABLE 2. Detailed parameters of the milling cutter.

Cutting tool	Teeth	Diameter	Helical angle	Total length	Length of cutting edge	Equality
EMC54120 4F	2	5	35°	75mm	32mm	142g

B. COMPARISON APPROACH

For the development of tool path for cavity milling, common development solutions such as offset loop cutting mode are often used to generate tool path, *e.g.* NX UG. In previous studies, the feasibility of evolutionary computation-based tool path generation solutions and their superiority in terms of generation speed and quality compared to common tool path development solutions have been demonstrated. It can be argued that the evolutionary computation-based tool path generation solution is overwhelmingly superior to both of these solutions. Therefore, in order to demonstrate the superiority of the RL-based tool path generation solution proposed in this paper, it is necessary to compare the proposed approach with the evolutionary computation-based solution. three commonly used evolutionary computation techniques are used as comparison approach.

(1) Genetic Algorithm (GA): GA is an individual-based optimisation algorithm. Its algorithm architecture consists of a selection mechanism, a crossover mechanism and a mutation mechanism. The selection mechanism is designed as a tournament mechanism, the crossover mechanism is designed as a two-point exchange mechanism and the variation mechanism is designed as a flipped slice mechanism. The population size is the same as the size of the discrete point set P, the crossover probability is set to 100% and the variation mechanism is set to 10%.

(2) Ant Colony Algorithm (ACO): ACO is a population-based optimisation algorithm. Its algorithmic architecture consists of multiple ants communicating with each other to achieve the optimisation process by virtue of the historical exploration experience of the solution space stored in the pheromone. The number of ants in the population is set to the same size as the discrete point set P, the pheromone importance factor is set to 1, the heuristic information importance factor is set to 5, the pheromone volatility factor is set to 0.10, and the constant factor used for the global pheromone update mechanism is set to 1.

(3) Particle Swarm Optimisation (PSO): PSO is a popular population optimisation algorithm. the optimisation process of PSO achieves the search for the optimal solution in the solution space through collaboration and information sharing among individuals in the population. the advantage of PSO is that fewer hyperparameters need to be set. The number of particles in the population is the same as the size of the discrete point set P.

For the RL-based approach proposed in this paper, the greedy factor ϵ of Q-learning algorithm is set as 0.10, the discount factor γ is set as 0.10 and the learning rate lr is set as 0.70. No extra hyperparameters need to be set in A* algorithm.

C. EXPERIMENTAL SETTING

The GA, ACO, PSO and RL-based approaches are used to complete the four given cavity milling tasks respectively. It is important to emphasise that the RL-based approach to TPO was originally designed (1) to solve TPO quickly and (2) to produce high quality results within the constraints of a short allowable running time. Therefore, it was necessary to stress test all algorithms in order to demonstrate that the RL-based approach performed as initially expected in terms of solution speed and solution quality. I set up five experiments with different stress levels. Specifically, we strictly limit the allowable running time of each algorithm to 3s, 30s, 60s, 5min and 10min. The allowable running time is the termination condition of the algorithm, and the cumulative running time of the program is checked immediately at the end of each iteration to see if the time limit is exceeded. If the time limit is exceeded, the algorithm is terminated, otherwise it continues to run. A shorter allowable running time means more pressure is placed on the algorithm. The purpose of setting a strong allowable time pressure is to demonstrate whether the algorithm has the ability to output a better solution in a short time. This capability is important for the two-layer optimisation model mentioned in Section I. This is because TPO is often used as a subproblem in two-layer optimisation models.

D. RESULT ANALYSIS

For workpieces 1, 2 and 3, the size of the discrete point set P is 610; for workpiece 4, the size of the discrete point set P is 1020. The problem sizes for all four experimental cases are large enough to say that none of them are trivial tasks. The results of the five sets of experiments are presented in Table 3-7.

The optimal approach in one case under one allowed running time setting is bolded. It can be observed that, in general, the RL-based approach significantly outperforms the evolutionary computation-based approach in different stress tests for different cases; specifically, when the allowed run time is 3s, the RL-based approach performs better on average by 19.28% on workpieces 1 to 4, using the other approaches as baseline, 31.08%, 46.49% and 12.47%. In total, the RL-based approach achieves a 27.33% performance advantage over the other approaches under this pressure. When allowed to run for 30s, the performance of the RL-based approach improves on average by 33.52%, 28.74%, 35.49% and 42.13% on artefacts 1 to 4, using the other approaches as a baseline. Overall, the RL-based approach achieves a performance advantage of 34.97% over the other approaches under this pressure. The algorithm was subject to a very loose time constraint when the allowed run time was 10 min. On artefacts 1 to 4, the performance of the RL-based approach improves on average by 45.26%, 31.86%, 43.47% and 58.87%, using the other approaches as a baseline. In total, the RL-based approach achieves a 44.87% performance advantage over the other approaches under this pressure. It can be said that in our experiments, the RL-based approach always performs well, regardless of the running time limit imposed on the algorithm. When the pressure is high (*i.e.,* the allowable running time is set short) the performance of all approaches is somewhat weakened and the RL-based approach outperforms the other approaches substantially. At lower pressures (*i.e.,* longer allowable run times) the performance of all approaches improves somewhat, but the RL-based approach still outperforms the other approaches by a large margin. It can be argued that the advantage of the RL-based approach lies not only in its ability to produce high quality solution outputs in short runs, but also in its ability to explore the solution space in long runs.

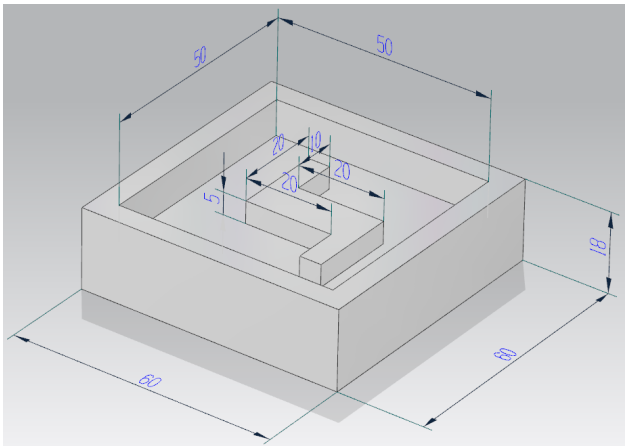

FIGURE 8. The 3D model with PMI dimensioning of the four test workpieces used in the experiment.

E. VISUALISATION EXPERIMENT

Given the large size of the four experimental cases used and the consequent high density of the planar grid of the cavity, it is not easy to illustrate the differences between the tool path output by the RL-based approach and the other comparative approach. Therefore, an additional experiment case of smaller size was designed to demonstrate the specific differences in the generated tool paths between the proposed approach and the comparative approach in order to visualise the characteristics and superiority of the tool path generated by the RL-based approach. The 3-dimensional

TABLE 3. Experimental results of the stress test (allowable running time: 3s).

	GA (3s)			ACO (3s)			PSO (3s)			RL (3s)		
	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v
Workpiece 1	621.00	1110.00	1731.00	477.00	966.00	1443.00	628.00	1120.00	1750.00	**432.00**	**921.00**	**1353.00**
Workpiece 2	679.00	1143.00	1822.00	478.00	942.00	1420.00	497.00	961.00	1458.00	**332.00**	**796.00**	**1128.00**
Workpiece 3	725.00	1197.00	1922.00	625.00	1097.00	1722.00	642.00	1114.00	1756.00	**344.00**	**816.00**	**1160.00**
Workpiece 4	1755.00	2653.00	4408.00	1580.00	2478.00	4058.00	1752.00	2650.00	4402.00	**1445.00**	**2343.00**	**3788.00**

TABLE 4. Experimental results of the stress test (allowable running time: 30s).

	GA (30s)			ACO (30s)			PSO (30s)			RL (30s)		
	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v
Workpiece 1	661.00	1150.00	1811.00	529.00	1018.00	1547.00	608.00	1097.00	1705.00	**363.00**	**852.00**	**1215.00**
Workpiece 2	646.00	1110.00	1756.00	393.00	857.00	1250.00	584.00	1048.00	1632.00	**341.00**	**805.00**	**1146.00**
Workpiece 3	677.00	1149.00	1826.00	593.00	1065.00	1658.00	585.00	1057.00	1642.00	**369.00**	**841.00**	**1210.00**
Workpiece 4	2019.00	2917.00	4936.00	1758.00	2656.00	4414.00	1837.00	2735.00	4572.00	**1104.00**	**2002.00**	**3106.00**

TABLE 5. Experimental results of the stress test (allowable running time: 60s).

	GA (60s)			ACO (60s)			PSO (60s)			RL (60s)		
	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v
Workpiece 1	552.00	1041.00	1593.00	448.00	937.00	1385.00	600.00	1089.00	1689.00	**412.00**	**901.00**	**1313.00**
Workpiece 2	697.00	1161.00	1858.00	509.00	973.00	1482.00	691.00	1155.00	1846.00	**339.00**	**803.00**	**1142.00**
Workpiece 3	531.00	1003.00	1534.00	528.00	1000.00	1528.00	535.00	1007.00	1542.00	**316.00**	**788.00**	**1104.00**
Workpiece 4	1910.00	2808.00	4718.00	1838.00	2736.00	4574.00	1875.00	2773.00	4648.00	**1257.00**	**2155.00**	**3412.00**

TABLE 6. Experimental results of the stress test (allowable running time: 5min).

	GA (5min)			ACO (5min)			PSO (5min)			RL (5min)		
	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v
Workpiece 1	597.00	1086.00	1683.00	500.00	989.00	1489.00	588.00	1077.00	1665.00	**419.00**	**908.00**	**1327.00**
Workpiece 2	697.00	1161.00	1858.00	536.00	1000.00	1536.00	496.00	960.00	1456.00	**253.00**	**717.00**	**970.00**
Workpiece 3	704.00	1176.00	1880.00	575.00	1047.00	1622.00	558.00	1030.00	1588.00	**411.00**	**883.00**	**1294.00**
Workpiece 4	1805.00	2703.00	4508.00	1783.00	2681.00	4464.00	1964.00	2862.00	4826.00	**1081.00**	**1979.00**	**3060.00**

TABLE 7. Experimental results of the stress test (allowable running time: 10min).

	GA (10min)			ACO (10min)			PSO (10min)			RL (10min)		
	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v	L_a	L_v	L_a+L_v
Workpiece 1	655.00	1144.00	1799.00	482.00	971.00	1453.00	568.00	1057.00	1625.00	**279.00**	**768.00**	**1047.00**
Workpiece 2	702.00	1166.00	1868.00	506.00	970.00	1476.00	515.00	979.00	1494.00	**345.00**	**809.00**	**1154.00**
Workpiece 3	693.00	1165.00	1858.00	532.00	1004.00	1536.00	638.00	1110.00	1748.00	**328.00**	**800.00**	**1128.00**
Workpiece 4	2051.00	2949.00	5000.00	1788.00	2686.00	4474.00	1990.00	2888.00	4878.00	**956.00**	**1854.00**	**2810.00**

model of the new experimental case is shown in Fig. 8. The GA, ACO, PSO and RL-based approaches were used to optimise the cavity milling task for the given model. As the aim of this experiment is to demonstrate the intuitive performance superiority of the RL-based approach over the comparative approach in tool path generation, the allowable running time of the algorithms will not be strongly constrained in this experiment. The allowable running time is set to 15 min for all algorithms.This is to ensure the reliability and fairness of the experimental results, and to avoid the suspicion that the performance of GA, ACO and PSO is suppressed by deliberately setting a short allowable running time.

The tool paths output by GA, ACO, PSO and RL-based approach are shown in Fig. 9.It can be seen that the RL-based approach outputs a very clean tool path without too many bidirectional arrows, which means that the blanking paths are shorter. The other approaches output a large number of bidirectional arrows, which means that they output a large number of meaningless and redundant blanking behaviour in the tool paths. Therefore, it can be said that the tool paths output by the RL-based approach are significantly

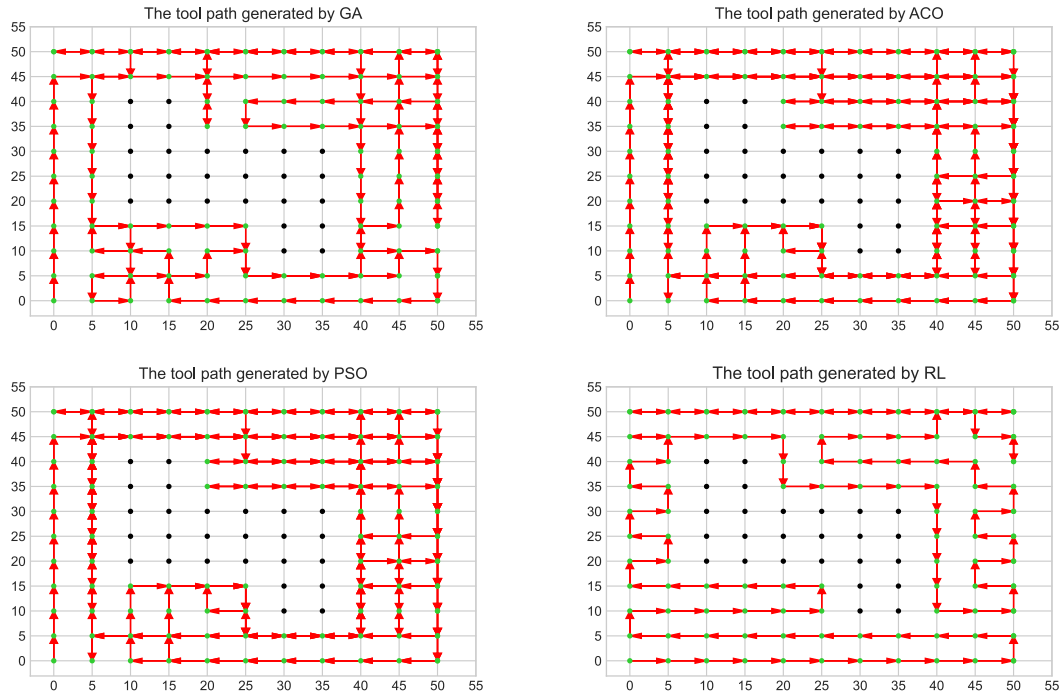

FIGURE 9. Tool path visualisation results from GA, PSO, ACO and RL-based approach outputs. Green dots indicate cuttable path points, black dots indicate non-cuttable path points. Red directional arrows indicate the cutting direction of the tool.

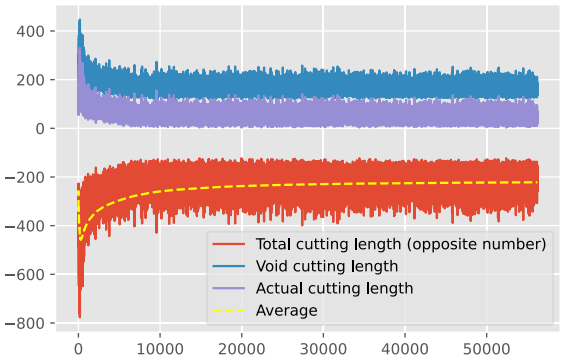

FIGURE 10. The training process of the RL-based approach.

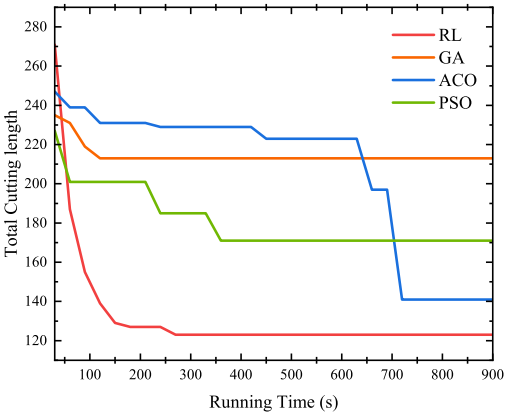

FIGURE 11. The iteration curves for GA, PSO, ACO and RL-based approach.

better than the tool paths output by the other compared approach.

Furthermore, Fig. 10 shows how the total tool length of the RL-based approach (*i.e.,* the negative of the sum of L_a and L_v) varies with training episodes. It can be seen that both the void cutting length, the actual cutting length and the total cutting length are continuously optimised with iterations of the episode.

In addition, in order to make a horizontal comparison with other comparison approach, Fig. 11 gives the convergence curves of GA, ACO, PSO and RL-based approach. In terms of convergence speed, it can be observed that the RL-based approach converges quickly to a better value, while the other compared approach either fall into a local optimum too early or converge late to a relatively better solution. In terms of solution quality, the RL-based approach is significantly better than GA, ACO and PSO, so we can conclude that the RL-based approach proposed in this paper is a better tool for solving TPO in terms of both solution speed and solution quality.

F. DISCUSSION

This section discusses the reasons why the proposed method outperforms the comparison methods. The experimental results clearly prove the superiority of the tool path planning algorithm based on reinforcement learning in terms of solution speed and solution quality. The proposed method not

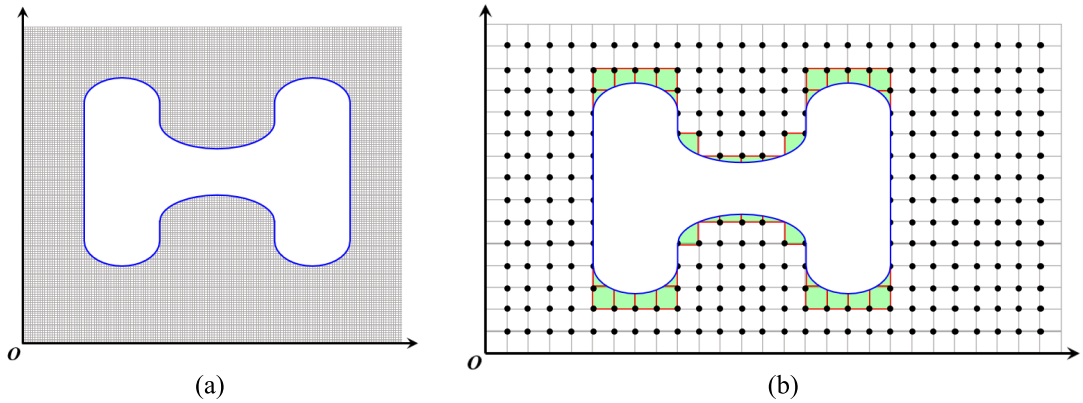

FIGURE 12. (a) Grid generation result for the curved contour; (b) Grid generation result using rectangles to approximate curve contour (the green area is the remaining metal after cutting).

only achieves better solutions in a very short running time compared to evolutionary computation based methods, but also outperforms many evolutionary computation algorithms in longer running times. These advantages of the proposed scheme are mainly attributed to the effective combination of reinforcement learning algorithm and A* algorithm which provides an effective supervision signal for the iterative optimization process. Specifically, in our proposed framework, the reinforcement learning algorithm is responsible for the coarse-grained tool path planning and the A* algorithm is responsible for the fine-grained tool path planning algorithm. In essence it is a two-level algorithm. The A* algorithm can quickly produce an optimized path from the starting point to the goal point, and its superior robustness in terms of the quality of the solution has been theoretically demonstrated. Therefore, the A* algorithm provides an efficient and stable supervision signal (*i.e.,* reward) to the agent (*i.e.,* the tool), which guides the agent to quickly explore the optimal machining path in the machining area. In contrast, the performance of solving schemes based on evolutionary computation completely depends on the information interaction and random mutation of population individuals in loop iterations. Such search models lack effective supervision signals in addition to the selection of the environment based on the fitness function. In addition, due to the randomness in the evolution process, the stability of its solution quality cannot be guaranteed.

In terms of algorithm efficiency, the solution search efficiency of the proposed method is much higher than that of the method based on evolutionary computation. According to the algorithm flow of evolutionary computation, its population update must be realized by environmental selection mechanism after all individuals have completed information exchange and mutation. Due to the low efficiency of evolutionary computation, it is difficult to obtain solutions with excellent performance in few iteration rounds. In contrast, reinforcement learning algorithms, whose search process is based on the relationship between actions and rewards, are able to effectively tune the experience (*i.e.,* the Q-table) multiple times within a single episode.

V. LIMITATIONS OF OUR MODEL

This section discusses the limitations of our proposed model. Theoretically, our proposed tool path generation framework is suitable for cavity milling tasks with complex contours, such as islands and holes with arcs or splines. This is because no matter what the contour of the cavity is, our method can be applied as long as its milling region is transformed into a 2D grid plane. However, when facing the contour edges of arbitrary splines, in case of overcutting, the grid interval of the grid converter mentioned in Section III-B needs to be modified to the minimum of the distance between the discrete points of the spline. Fig. 12(a) shows the 2D grid generation results in which the cavity region is a curved contour structure. It can be found that once the grid interval is set to the minimum of the distance between the spline discrete points, it results in a large increase in the number of grid points. This will severely increase the computational burden of the algorithm. Therefore, when dealing with cavity milling scenarios with complex contours, we recommend to use rectangles to approximate curve contours, as shown in Fig. 12(b). In this way, not only the number of grid points is reduced and thus the computational cost is reduced, but also the proposed framework can be directly used without modifying the grid interval. The only drawback of this approach is the need to cut the remaining metal around the contour.

VI. CONCLUSION

In this paper, the problem of tool path optimisation for complex cavity milling is well defined and investigated. The proposed approach consists of a grid converter and an RL-based tool path generator. The grid converter is used to transform a given 3D cavity model into a grid plane consisting of discrete tool path points, which is further used as the environment for the RL-based tool path generator. the Q-learning algorithm is used as a coarse-grained path planning approach and the A* algorithm as a fine-grained path planning approach. Modelling of TPO using MDP theory theoretically guarantees the deployment of the Q-learning algorithm. Numerical

experiments demonstrate that the proposed approach significantly outperforms evolutionary computation-based approach in terms of solution speed and solution quality. The advantages of the proposed approach in terms of solution speed and solution quality can empower the efficiency of TPO in two-layer optimisation models, and indirectly improve the efficiency and quality of solutions for two-layer optimisation models, as TPO is often used as a sub-problem embedded in the overall model. Therefore, it is necessary to deploy the TPO solution proposed in this paper into the two-layer optimisation model as soon as possible in order to optimise the manufacturing system as a whole and not only for the tool path.

REFERENCES

[1] W. Fuzhong, "Optimal generation method of NC milling tool path for pocket with island," in *Proc. Int. Conf. Mech. Autom. Eng.*, Jul. 2013, pp. 73–75.
[2] M. Zhang, W.-J. Liu, and G.-C. Yang, "Complex pocket milling region automatic identification method," *China Mech. Eng.*, vol. 21, no. 18, p. 2224, 2010.
[3] Z. Yingjie and Z. Liujie, "Tool path optimization to regulate the cutting forces in pocket machining," in *Proc. 4th IEEE Conf. Ind. Electron. Appl.*, May 2009, pp. 2764–2769.
[4] H.-C. Kim, S.-G. Lee, and M.-Y. Yang, "An optimized contour parallel tool path for 2D milling with flat endmill," *Int. J. Adv. Manuf. Technol.*, vol. 31, nos. 5–6, pp. 567–573, Nov. 2006.
[5] G. Zhou, C. Zhang, F. Lu, and J. Zhang, "Integrated optimization of cutting parameters and tool path for cavity milling considering carbon emissions," *J. Cleaner Prod.*, vol. 250, Mar. 2020, Art. no. 119454.
[6] K. Xu, M. Luo, and K. Tang, "Machine based energy-saving tool path generation for five-axis end milling of freeform surfaces," *J. Cleaner Prod.*, vol. 139, pp. 1207–1223, Dec. 2016.
[7] X. Zhao, H. Zhao, S. Wan, X. Li, and H. Ding, "An analytical decoupled corner smoothing method for five-axis linear tool paths," *IEEE Access*, vol. 7, pp. 22763–22772, 2019.
[8] J. Barclay, V. Dhokia, and A. Nassehi, "Generating milling tool paths for prismatic parts using genetic programming," *Proc. CIRP*, vol. 33, pp. 490–495, Jan. 2015.
[9] R. K. Agrawal, D. K. Pratihar, and A. R. Choudhury, "Optimization of CNC isoscallop free form surface machining using a genetic algorithm," *Int. J. Mach. Tools Manuf.*, vol. 46, nos. 7–8, pp. 811–819, Jun. 2006.
[10] A. Petunin, "Tools path optimization for CNC cutting machines," *Vestnik UGATU Syst. Eng. Inf. Technol.*, vol. 15, no. 4, p. 44, 2011.
[11] N. Medina-Rodríguez, O. Montiel-Ross, R. Sepúlveda, and O. Castillo, "Tool path optimization for computer numerical control machines based on parallel ACO," *Eng. Lett.*, vol. 20, no. 1, 2012.
[12] N. Hatem, Y. Yusof, A. Z. A. Kadir, K. Latif, and M. A. Mohammed, "A novel integrating between tool path optimization using an ACO algorithm and interpreter for open architecture CNC system," *Expert Syst. Appl.*, vol. 178, Sep. 2021, Art. no. 114988.
[13] K.-Y. Fok, C.-T. Cheng, N. Ganganath, H. H.-C. Iu, and C. K. Tse, "An ACO-based tool-path optimizer for 3-D printing applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2277–2287, Apr. 2019.
[14] M. Amersdorfer and T. Meurer, "Equidistant tool path and Cartesian trajectory planning for robotic machining of curved freeform surfaces," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3311–3323, Oct. 2022.
[15] N. Sugita, T. Nakano, T. Kato, Y. Nakajima, and M. Mitsuishi, "Tool path generator for bone machining in minimally invasive orthopedic surgery," *IEEE/ASME Trans. Mechatronics*, vol. 15, no. 3, pp. 471–479, Jun. 2010.
[16] J. Yi, C.-H. Chu, C.-L. Kuo, X. Li, and L. Gao, "Optimized tool path planning for five-axis flank milling of ruled surfaces using geometric decomposition strategy and multi-population harmony search algorithm," *Appl. Soft Comput.*, vol. 73, pp. 547–561, Dec. 2018.
[17] L. Li, X. Deng, J. Zhao, F. Zhao, and J. W. Sutherland, "Multi-objective optimization of tool path considering efficiency, energy-saving and carbon-emission for free-form surface milling," *J. Clean. Prod.*, vol. 172, pp. 3311–3322, Jan. 2018.
[18] N. Baranov and A. Feofanov, "Tool path optimization in complex geometry parts machining process," *Mater. Today, Proc.*, vol. 38, pp. 1364–1366, Jan. 2021.
[19] N. A. Fountas and N. M. Vaxevanidis, "Intelligent 3D tool path planning for optimized 3-axis sculptured surface CNC machining through digitized data evaluation and swarm-based evolutionary algorithms," *Measurement*, vol. 158, Jul. 2020, Art. no. 107678.
[20] Z. Khodabakhshi and A. Hosseini, "Optimization of non-productive tool path in drilling: A review," *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 426–431, 2021.
[21] Q. Guo, Y. Sun, Y. Jiang, Y. Yan, B. Zhao, and P. Ming, "Tool path optimization for five-axis flank milling with cutter runout effect using the theory of envelope surface based on CL data for general tools," *J. Manuf. Syst.*, vol. 38, pp. 87–97, Jan. 2016.
[22] L. Ying, M. Li, and J. Yang, "Agglomeration and driving factors of regional innovation space based on intelligent manufacturing and green economy," *Environ. Technol. Innov.*, vol. 22, May 2021, Art. no. 101398.
[23] X. Yang, Y. He, R. Liao, Y. Cai, and J. Ai, "Integrated mission reliability modeling based on extended quality state task network for intelligent multistate manufacturing systems," *Rel. Eng. Syst. Saf.*, vol. 223, Jul. 2022, Art. no. 108495.
[24] T. Zheng, Y. Liang, B. Wang, H. Sun, J. Zheng, D. Li, Y. Chen, L. Shao, and H. Zhang, "A two-stage improved genetic algorithm-particle swarm optimization algorithm for optimizing the pressurization scheme of coal bed methane gathering networks," *J. Cleaner Prod.*, vol. 229, pp. 941–955, Aug. 2019.
[25] R. Yan, J. Wang, J. Wang, L. Tian, S. Tang, Y. Wang, J. Zhang, Y. Cheng, and Y. Li, "A two-stage stochastic-robust optimization for a hybrid renewable energy CCHP system considering multiple scenario-interval uncertainties," *Energy*, vol. 247, May 2022, Art. no. 123498.
[26] Y. Huang, Y. Wang, and N. Liu, "A two-stage energy management for heat-electricity integrated energy system considering dynamic pricing of Stackelberg game and operation strategy optimization," *Energy*, vol. 244, Apr. 2022, Art. no. 122576.
[27] S. Gong, C. Shao, and L. Zhu, "Energy efficiency enhancement of energy and materials for ethylene production based on two-stage coordinated optimization scheme," *Energy*, vol. 217, Feb. 2021, Art. no. 119401.
[28] H. Sun, Y. Ge, W. Liu, and Z. Liu, "Geometric optimization of two-stage thermoelectric generator using genetic algorithms and thermodynamic analysis," *Energy*, vol. 171, pp. 37–48, Mar. 2019.
[29] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment," *IEEE Access*, vol. 9, pp. 59196–59210, 2021.
[30] Z. He, C. Liu, X. Chu, R. R. Negenborn, and Q. Wu, "Dynamic anti-collision A-star algorithm for multi-ship encounter situations," *Appl. Ocean Res.*, vol. 118, Jan. 2022, Art. no. 102995.
[31] S. Lim, H. Yu, and H. Lee, "Optimal tethered-UAV deployment in A2G communication networks: Multi-agent Q-Learning approach," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 18539–18549, Oct. 2022.
[32] P. C. Ng and J. She, "Remote proximity sensing with a novel *Q*-learning in Bluetooth low energy network," *IEEE Trans. Wireless Commun.*, vol. 21, no. 8, pp. 6156–6166, Aug. 2022.
[33] Y. Xu, Z. Zhao, and S. Yin, "Performance optimization and fault-tolerance of highly dynamic systems via *Q*-learning with an incrementally attached controller gain system," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 15, 2022, doi: 10.1109/TNNLS.2022.3155876.
[34] C. C. White III and J. D. White, "Markov decision processes," *Eur. J. Oper. Res.*, vol. 39, no. 1, pp. 1–16, Mar. 1989.
[35] H. Okamura, S. Miyata, and T. Dohi, "A Markov decision process approach to dynamic power management in a cluster system," *IEEE Access*, vol. 3, pp. 3039–3047, 2015.
[36] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019.
[37] J. A. Boyan, "Technical update: Least-squares temporal difference learning," *Mach. Learn.*, vol. 49, nos. 2–3, pp. 233–246, 2002.

[38] Y. Su, R. Fan, X. Fu, and Z. Jin, "DQELR: An adaptive deep *Q*-network-based energy- and latency-aware routing protocol design for underwater acoustic sensor networks," *IEEE Access*, vol. 7, pp. 9091–9104, 2019.
[39] A. Iqbal, M.-L. Tham, and Y. C. Chang, "Double deep *Q*-network-based energy-efficient resource allocation in cloud radio access network," *IEEE Access*, vol. 9, pp. 20440–20449, 2021.
[40] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Reinforcement learning through asynchronous advantage actor-critic on a GPU," 2016, *arXiv:1611.06256*.
[41] F. F. M. El-Sousy and F. A. F. Alenizi, "Optimal adaptive super-twisting sliding-mode control using online actor-critic neural networks for permanent-magnet synchronous motor drives," *IEEE Access*, vol. 9, pp. 82508–82534, 2021.

YI WAN (Member, IEEE) received the Ph.D. degree in materials science and engineering from the Nanjing University of Science and Technology, Nanjing, in 2008. She is currently an Associate Professor with the School of Environmental Science, Nanjing Xiaozhuang University, Nanjing. Her research interests include the analysis of tribological systems, simulation planning and design, and innovation approach.

WEI XU is currently a Senior Experimentalist with the School of Mechanical and Electrical Engineering, Sanjiang University, Nanjing, China. His current research interests include intelligence manufacturing, optimal design, and virtual simulation.

TIAN-YU ZUO is currently pursuing the M.S. degree with the School of Automation, Nanjing University of Information Science and Technology, Nanjing, China.

His works have been published in IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS and *Applied Soft Computing*. His current research interests include statistical machine learning, transfer learning, reinforcement learning, evolutionary algorithms, swarm intelligence, and their applications in real-world optimization problems and climate projection.