DEMA-3D TSP: An Enhanced Reinforcement Learning with DEMA Attention in Sequence Optimization for Safflower Picking Robot

LI Menghao[1], WANG Xiaorong[1,2,3*], LIU Zihe[1], DUAN Mengyu[1], JIN Zhengyang[1]

(1. *School of Mechanical Engineering*, *Xinjiang University*, *Urumqi* 830017, *China*; 2. *Agriculture and Animal Husbandry Robot and Intelligent Equipment Engineering Research Center of Xinjiang Uygur Autonomous Region*, *Urumqi* 830017, *China*; 3. *Engineering Training Center of Xinjiang University*, *Urumqi* 830017, *China*)

Abstract: [Objective] There are several critical challenges in automated safflower harvesting, particularly the inefficiencies in path planning, suboptimal route quality, and limited decision-making capability under dynamic and complex environments. To solve these issues, the problem was formulated as a three-dimensional traveling salesman problem and an enhanced reinforcement learning model named actor-critic reinforcement learning pointer network (AC-RL-PtrNet) was proposed, specifically designed for deployment on intelligent safflower picking robots in agricultural settings. **[Methods]** First, to address the inherent limitations of conventional attention mechanisms in dynamic environments with complex spatial structures, an enhanced attention module was proposed based on the dynamic exponential moving average framework. By combining multi-head attention, spatial distance encoding, and adaptive exponential smoothing, the improved design allowed the model to better capture long-range dependencies and spatial context among safflowers. Meanwhile, to minimize computational cost while preserving inference quality, a structured pruning approach was adopted, which selectively removed redundant connections in the long short-term memory gates and fully connected layers. In parallel, the critic network was redesigned to improve learning stability and accuracy. This was achieved through the inclusion of batch normalization, residual feature aggregation, and a multi-layer value estimation head, all of which contributed to a tighter actor-critic synergy during policy training. **[Results and Discussions]** To quantitatively assess the impact of each component, ablation experiments were conducted across various configurations. The results confirmed that each module contributed distinct benefits, while their combination yielded the highest improvements in both planning precision and inference efficiency. This coordinated actor-critic design effectively enhanced both trajectory quality and decision stability, which were critical in sequential robotic picking tasks. Experimental results also demonstrated that, compared with traditional swarm intelligence algorithms particle swarm optimization (PSO), ant colony optimization (ACO), and non-dominated sorting genetic algorithm, the proposed AC-RL-PtrNet model achieved a planning time improvement ranging from −2.63% to 61.87% on the 25-target dataset and from 22.93% to 59.1% on the 31-target dataset. Meanwhile, the optimized paths were significantly shortened across different planning instances, indicating robust generalization capability under varied problem scales. Furthermore, field experiments provided concrete validation of the model's practical applicability. When deployed on a mobile picking robot in real safflower fields, the AC-RL-PtrNet achieved a 9.56% reduction in path length and 5.43% time saved for a 25-target picking task, and a 20.17% path reduction and 29.70% time saving for a 31-target scenario involving a different safflower variety. Overall, these results all indicated that the proposed method exhibited significant advantages in enhancing path planning efficiency and optimizing path quality. **[Conclusions]** This study offers a practical solution for achieving efficient and robust automatic picking by safflower picking robots and provides new insights into solving 3D combinatorial optimization problems.

Received date: 2025-05-19

Foundation items: Natural Science Foundation of Xinjiang Uygur Autonomous Region, China Under Grant (2023D01C190); National Science and Technology Major Project (2022ZD0115801)

First author: LI Menghao, E-mail: LiMenghao@stu.xju.edu.cn

*Corresponding author: WANG Xiaorong, E-mail: xiaorongwang@xju.edu.cn

Keywords: dynamic exponential moving average mechanism; structural pruning; reinforcement learning; 3D traveling salesman problem; safflower picking; robot
CLC number: S238 **Documents code: A** **Article ID: SA202506004**

Citation: LI Menghao, WANG Xiaorong, LIU Zihe, DUAN Mengyu, JIN Zhengyang. DEMA-3D TSP: An Enhanced Reinforcement Learning with DEMA Attention in Sequence Optimization for Safflower Picking Robot[J]. Smart Agriculture, 2026, 8(2): 200-219. DOI: 10.12133/j.smartag.SA202506004 (in English with Chinese abstract)
李萌浩, 王小荣, 刘子贺, 段孟渝, 金正阳 . DEMA-3D TSP: 一种应用在红花采摘机器人序列规划的融合DEMA注意力机制强化学习算法[J]. 智慧农业(中英文), 2026, 8(2): 200-219. DOI: 10.12133/j.smartag.SA202506004

0 Introduction

Safflower (*Carthamus tinctorius* L.), a distinctive economic crop with medicinal, culinary, and industrial value, has witnessed a significant increase in its market value[1-3]. As one of the primary producers of safflower, China, particularly the Xinjiang Uygur Autonomous Region (Xinjiang region), has emerged as a core supply center[4, 5]. To enable efficient picking within the plant's growth cycle, mechanized picking has become a prominent focus of research[6-8]. In particular, the application of robotic technologies offers promising solutions for the automated picking of safflower filament[9].

To address the sequencing problem encountered in the picking point planning process of safflower picking robots, it is essential to recognize its inherent similarity to the traveling salesman problem (TSP). The TSP aims to determine the shortest path that visits a set of designated locations exactly once before returning to the starting point[10-12]. Drawing on this principle, a combinatorial optimization model based on the three-dimensional traveling salesman problem (3D TSP) was developed for safflower picking. Traditionally, swarm intelligence algorithms have been widely employed to solve TSP-related problems[13, 14]. UTAMIMA and REINERS[15] proposed a hybrid algorithm combining genetic algorithm (GA), tabu search (TS), and ant colony optimization (ACO), which effectively addressed the path planning problem in multi-field and multi-machine agricultural scenarios. The method reduced travel distance by an average of 16.21% compared to traditional approaches and significantly shortened computation time. GAO et al.[16] proposed an improved particle swarm optimization (PSO) for apple picking robot path planning, which enhanced obstacle avoidance and significantly improved picking efficiency. FANG et al.[17] proposed a spraying path planning method based on the vector modeling method, which effectively ensures full boundary coverage while minimizing redundant overlaps and turning points. This approach significantly improved the operational efficiency and coverage precision of unmanned helicopter pesticide spraying tasks. These algorithms simulated the collective behavior mechanisms observed in nature, enabling them to find near optimal solutions. However, as the number of individuals increases, swarm intelligence algorithms must recompute the path from scratch for each iteration. This leads to low computational efficiency in dynamic environments, rendering them less suitable for safflower picking tasks.

In contrast, reinforcement learning (RL) offers the advantage of offline learning[18,19]. By training on simulated data, RL models can acquire efficient path planning strategies[20]. In the context of safflower picking, RL models can rapidly adapt to changes in the number of target flowers, thereby enhancing picking efficiency, flexibility, and robustness. For example, ZHANG et al. [21] proposed a model-based temporal proximity soft actor-critic algorithm, integrating a long short-term memory (LSTM) model to optimize the maize threshing process by enhancing threshing quality and reducing damage rates. YANG et al.[22] proposed a path planning method for agricultural robots based on an improved RL approach. By integrating a residual network structure with a multi-step temporal difference error mechanism, the method significantly improved real-time decision-making capabilities in dynamic environments. However, its collaborative optimization of perception and obstacle avoidance in three-dimensional complex terrains, such as densely planted crop areas, requires further investigation. LIN et al.[23] developed a path planning framework based on recurrent deep deterministic policy gradient, in which an LSTM structure was used to retain historical state information, and continuous action outputs enabled collision free path generation. This method demonstrated superior obstacle avoidance and path stability under dynamic conditions, although its computational com-

plexity remains relatively high, leaving room for optimization in large-scale picking scenarios. Despite progress in applying deep reinforcement learning to agricultural path planning, several challenges persist in handling complex and dynamic environments, including inadequate state representation, insufficient real time responsiveness, and low training efficiency.

To address these issues, researches have recently begun integrating pointer network architectures with RL[24]. The RL pointer network processes input sequences directly and outputs optimized decision sequences. This not only improves state representation but also leverages offline learning to reduce computational costs during online inference. BELLO et al.[25] were the first who introduced RL via policy gradient methods into pointer networks to TSP in an end-to-end, unsupervised learning setting. GU and YANG[26] proposed a deep learning framework based on pointer networks trained with a hybrid strategy combining supervised learning and reinforcement learning to solve the NP-hard Max-Cut problem. Their results demonstrated that deep neural networks can effectively approximate solutions to large-scale combinatorial optimization problems. However, applying such models directly to safflower picking where target distributions change rapidly still suffers from suboptimal real-time decision making efficiency, primarily due to excessive search space and the resulting computational latency. To address this, LIN et al.[27] proposed an enhanced RL pointer network structure with a refined sequence planning strategy, which effectively reduced the computational burden in high-density environments and improved the rationality of picking sequences. Compared to traditional optimization methods, their approach reduced the average path length by 11.3% and achieved a 2.4× improvement in inference speed, highlighting the efficiency advantages of reinforcement learning in complex agricultural scenarios. Nevertheless, due to the unique growth patterns and picking requirements of safflower, real-world applications still face challenges such as dynamically changing flower distributions and complex picking paths. Although the aforementioned studies have progressively enhanced the performance of RL-based pointer networks in combinatorial optimization tasks, key limitations remain: Excessive path redundancy in dynamic environments, limited spatial search capacity of conventional pointer networks, reduced decision-making efficiency, and prolonged inference time. These issues are particularly critical in robotic safflower picking, where the spatial layout of targets is non-uniform and decision latency directly impacts task success. Moreover, insufficient coordination between the actor and critic networks can lead to unstable training and suboptimal policy convergence.

To address the aforementioned challenges, an actor-critic reinforcement learning pointer network algorithm (AC-RL-PtrNet) was developed for safflower picking scenarios. In this framework, three main enhancements were designed. First, an enhanced dynamic exponential moving average (DEMA)-based attention module with distance encoding was introduced, enabling more effective modeling of both local and global spatial structures among safflower targets. Second, a structured pruning strategy was applied to LSTM gates and dense layers to reduce network redundancy and inference latency, while preserving decision performance under dense or dynamically changing picking configurations. Third, the critic network was restructured with input normalization, compact nonlinear layers, and residual connections, allowing more stable value estimation and improved actor-critic optimization during policy learning in complex field environments.

1 Materials and methods

1.1 Data preparation

1.1.1 Safflower picking robot and working principle

In this study, a safflower picking robot based on a parallel robotic arm was employed. The system consists of a high-clearance mobile chassis, a parallel robotic arm, an end effector, a negative pressure safflower filament collection module, a visual recognition system, and a control system. The overall dimensions of the robot are 1 600 mm×1 280 mm×1 400 mm (length×width×height), and the operational workspace of the parallel robotic arm is defined by a cylindrical volume with a radius of 300 mm and a height of 300 mm. The working principle of the parallel safflower picking robot is illustrated in Fig. 1. During each picking task, the robot first identifies and localizes the safflower filament using its vision system and image processing algorithms[28]. The detected coordinates are then transformed from the camera coordinate frame to

the robotic arm coordinate frame. Subsequently, the optimal sequence is determined using a path planning algorithm. Finally, as shown in Fig. 1c and Fig. 1d, the robotic arm follows the planned trajectory under the control of the onboard system. The end effector, in conjunction with the negative pressure safflower filament collection module, performs the picking and collection of the safflower filament.

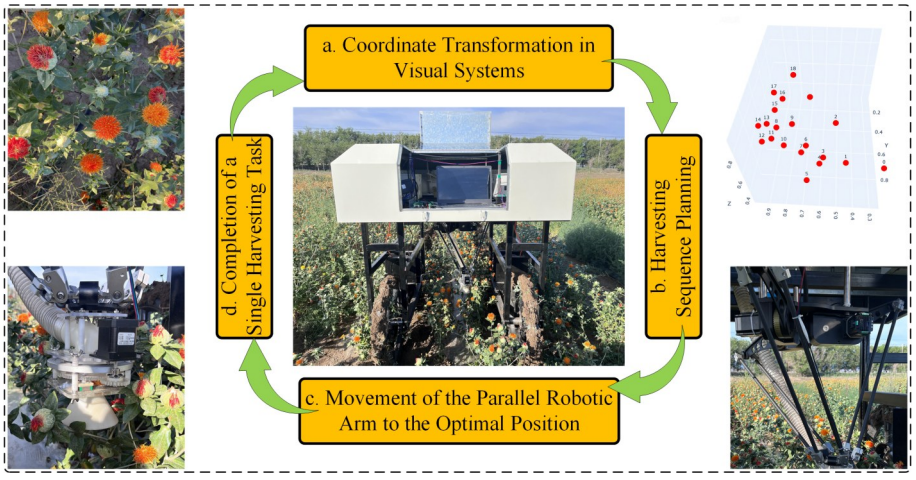


Fig. 1　Flowchart of safflower picking by parallel robot

1.1.2　Safflower data

During the safflower picking season from July to September, a field survey was conducted in Jimusaer county, Xinjiang. The investigation revealed that safflower is cultivated in rigid rows, with an inter-row spacing of 600 mm and an intra-row plant spacing of 150 mm, respectively. Within the reachable workspace of a single robotic arm, the number of safflower capitula was statistically found to range between 31 and 50. Based on this, 40 flower heads were chosen as a representative reference for training and evaluation.

Table 1 to Table 3 present the spatial distributions and corresponding coordinate points of 25, 31, and 43 safflower targets, respectively. To construct the dataset, a binocular stereo vision system mounted on an actual safflower-picking robot was employed. As illustrated in Fig. 2, the system first captured RGB images of safflower plants under field conditions. These included scenarios with both occluded and non-occluded safflower heads. The image recognition and localization algorithm then processed the images and computed the corresponding 3D coordinates of each detected safflower head. The recorded values represent the three-dimensional positions (X, Y, Z) of the safflower heads relative to the stereo camera's reference frame. To enhance computational efficiency and enable more stable learning during model training and inference, all spatial coordinates were further normalized to a $[0,1]$ range after acquisition. This normalization ensures consistent input scaling and facilitates the decision-making process of the reinforcement learning model. The resulting dataset serves as a real-world test set that simulates irregularly distributed picking targets, providing a robust benchmark for evaluating the proposed path planning algorithm under complex spatial conditions.

1.2　Combinatorial optimization problem formulation

Input definition: For the three-dimensional traveling salesman problem (3D-TSP), the input is defined as a set of 3D coordinates representing the positions of safflower filaments, denoted as $X=\{x_1,x_2,\cdots,x_N\}$, where $x_i=(x_i^1,x_i^2,x_i^3)\in\mathbb{R}^3$ indicates the spatial location of the i-th safflowers.

Output definition: The model aims to output a visiting sequence $\pi=(\pi_1,\pi_2,\cdots,\pi_N)$, which satisfies the Hamiltonian circuit constraint , that is, each point is visited exactly once and minimizes the total tour length, defined as Equation (1):

$$L(\pi)=\sum_{t=1}^{N}\| x_{\pi_t}-x_{\pi_{t+1}}\|^2 \tag{1}$$

Where, $L(\pi)$ denotes the total tour length; N is the total number of points.

Fig. 3 provides a detailed illustration of the process in which the safflower data, serving as the problem input, is processed by a RL pointer network model based on the actor-critic architecture, ultimately resulting in the output of the picking sequence.

Table 1 Spatial distribution and 3D coordinates of 25 safflower targets

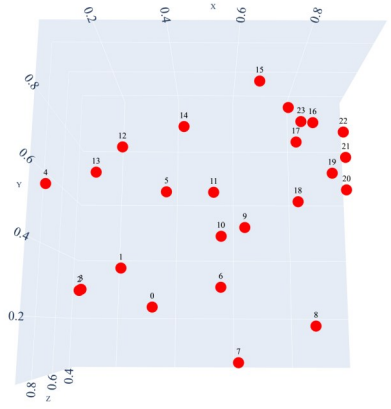
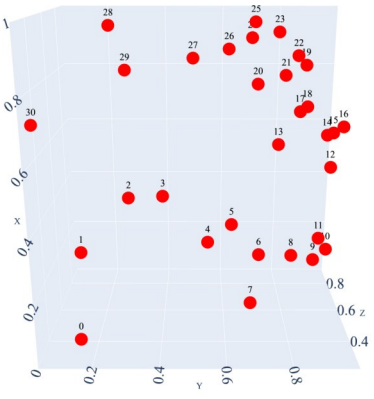
Spatial distribution	Label	Coordinate points	Label	Coordinate points
	1	(0.370 85, 0.225 16, 0.845 60)	14	(0.107 30, 0.718 10, 0.351 90)
	2	(0.2607 1, 0.3415 2, 0.684 10)	15	(0.444 20, 0.735 80, 0.791 80)
	3	(0.150 57, 0.274 46, 0.719 60)	16	(0.668 30, 0.909 50, 0.713 80)
	4	(0.083 69, 0.296 23, 0.402 10)	17	(0.829 70, 0.785 10, 0.691 60)
	5	(0.091 51, 0.540 71, 0.899 40)	18	(0.773 30, 0.714 30, 0.719 50)
	6	(0.407 40, 0.517 10, 0.910 40)	19	(0.794 20, 0.560 00, 0.591 80)
	7	(0.549 00, 0.278 40, 0.820 40)	20	(0.896 50, 0.643 30, 0.628 80)
	8	(0.608 00, 0.065 50, 0.524 90)	21	(0.942 30, 0.594 00, 0.613 50)
	9	(0.820 50, 0.177 80, 0.694 80)	22	(0.938 50, 0.692 60, 0.634 90)
	10	(0.619 90, 0.457 80, 0.710 90)	23	(0.968 60, 0.824 70, 0.490 60)
	11	(0.551 69, 0.434 36, 0.694 90)	24	(0.841 50, 0.897 50, 0.391 80)
	12	(0.529 40, 0.564 48, 0.701 90)	25	(0.801 60, 0.962 40, 0.361 80)
	13	(0.284 33, 0.649 31, 0.872 60)		

Table 2 Spatial distribution and 3D coordinates of 31 safflower targets

Spatial distribution	Label	Coordinate points	Label	Coordinate points
	1	(0.031 29, 0.153 36, 0.352 20)	17	(0.670 69, 0.953 60, 0.498 40)
	2	(0.195 88, 0.113 25, 0.634 10)	18	(0.676 65, 0.848 85, 0.684 30)
	3	(0.345 74, 0.266 43, 0.769 10)	19	(0.701 89, 0.865 28, 0.642 40)
	4	(0.417 69, 0.398 49, 0.586 40)	20	(0.797 98, 0.896 00, 0.891 50)
	5	(0.170 80, 0.548 25, 0.781 10)	21	(0.766 78, 0.708 47, 0.688 10)
	6	(0.273 79, 0.629 54, 0.697 40)	22	(0.812 71, 0.784 00, 0.581 70)
	7	(0.100 57, 0.739 19, 0.826 40)	23	(0.853 47, 0.841 81, 0.724 50)
	8	(0.024 39, 0.693 11, 0.597 60)	24	(0.944 39, 0.759 25, 0.573 90)
	9	(0.092 73, 0.861 65, 0.837 10)	25	(0.919 31, 0.683 73, 0.649 20)
	10	(0.051 98, 0.953 60, 0.887 40)	26	(0.968 68, 0.692 05, 0.637 40)
	11	(0.187 26, 0.958 50, 0.682 74)	27	(0.878 55, 0.613 11, 0.695 20)
	12	(0.264 38, 0.914 77, 0.599 40)	28	(0.852 53, 0.497 48, 0.673 30)
	13	(0.473 02, 0.967 90, 0.726 40)	29	(0.971 19, 0.255 98, 0.488 40)
	14	(0.527 73, 0.798 08, 0.822 40)	30	(0.812 71, 0.277 10, 0.683 30)
	15	(0.582 28, 0.953 60, 0.742 70)	31	(0.730 42, 0.069 73, 0.268 40)
	16	(0.599 52, 0.966 62, 0.705 60)		

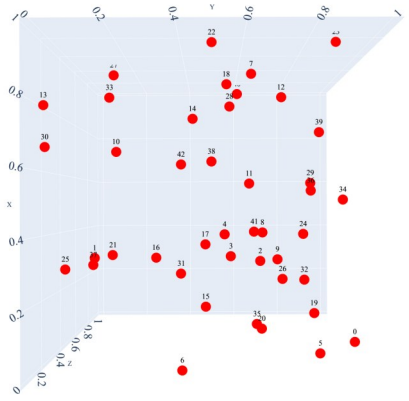
1.3 Overview of the RL pointer network based on the actor−critic architecture

In solving combinatorial optimization problems, the integration of RL with pointer networks has demonstrated powerful modeling and decision-making capabilities. This approach adopts an end-to-end learning paradigm, enabling the autonomous exploration of high-quality path planning strategies in unknown environments. This section sequentially presents the LSTM based encoder-decoder architecture, the formulation of the Markov decision process (MDP), the policy gradient method, and the specific implementation of the actor-critic dual network collaborative optimization within this framework.

1.3.1 Encoder−decoder architecture

1) Encoder: Spatiotemporal feature extraction with LSTM traditional sequence modeling methods struggle to capture complex spatial relationships in three-dimensional space. In this study, an LSTM network was employed to process the input state se-

Table 3 Spatial distribution and 3D coordinates of 43 safflower targets

Spatial distribution	Label	Coordinate points	Label	Coordinate points
	1	(0.066 90, 0.944 80, 0.220 69)	23	(0.958 28, 0.501 20, 0.077 57)
	2	(0.272 09, 0.007 86, 0.789 90)	24	(0.978 76, 0.869 70, 0.144 32)
	3	(0.291 00, 0.675 90, 0.498 20)	25	(0.413 78, 0.755 43, 0.050 40)
	4	(0.274 50, 0.580 01, 0.828 09)	26	(0.280 54, 0.011 32, 0.336 20)
	5	(0.387 98, 0.546 18, 0.486 30)	27	(0.236 89, 0.745 04, 0.409 20)
	6	(0.026 03, 0.840 90, 0.239 20)	28	(0.970 50, 0.141 88, 0.509 40)
	7	(0.017 18, 0.414 45, 0.101 10)	29	(0.881 24, 0.569 85, 0.649 90)
	8	(0.927 67, 0.631 05, 0.318 80)	30	(0.582 91, 0.902 41, 0.734 50)
	9	(0.371 76, 0.722 19, 0.901 90)	31	(0.662 84, 0.021 60, 0.092 40)
	10	(0.269 97, 0.768 43, 0.741 02)	32	(0.204 40, 0.369 01, 0.800 06)
	11	(0.725 02, 0.085 77, 0.862 82)	33	(0.169 60, 0.898 20, 0.854 90)
	12	(0.559 95, 0.614 29, 0.183 17)	34	(0.839 21, 0.173 10, 0.265 33)
	13	(0.884 94, 0.752 67, 0.490 53)	35	(0.513 48, 0.998 80, 0.585 30)
	14	(0.783 94, 0.016 53, 0.096 84)	36	(0.033 25, 0.672 79, 0.615 30)
	15	(0.787 99, 0.435 66, 0.371 51)	37	(0.549 80, 0.890 58, 0.660 40)
	16	(0.122 35, 0.477 45, 0.504 56)	38	(0.298 60, 0.111 27, 0.308 30)
	17	(0.297 22, 0.291 65, 0.551 51)	39	(0.636 40, 0.499 50, 0.296 30)
	18	(0.321 45, 0.471 07, 0.879 74)	40	(0.824 54, 0.998 80, 0.981 20)
	19	(0.831 73, 0.542 33, 0.048 44)	41	(0.971 44, 0.608 50, 0.847 90)
	20	(0.105 04, 0.866 72, 0.470 37)	42	(0.372 60, 0.688 90, 0.952 72)
	21	(0.048 52, 0.678 90, 0.475 20)	43	(0.611 60, 0.412 78, 0.091 30)
	22	(0.344 70, 0.201 90, 0.167 13)		

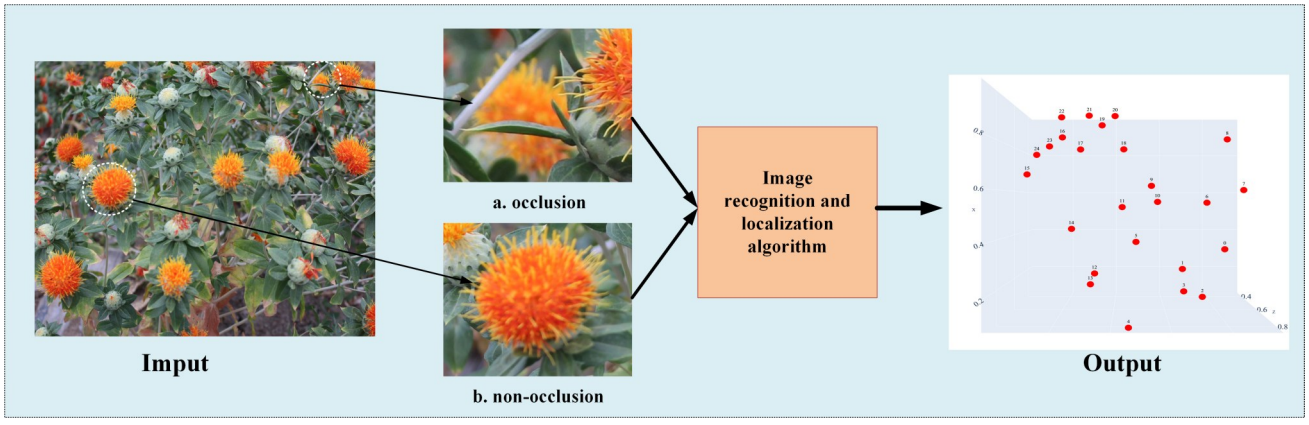


Fig. 2 Workflow of vision-based safflower recognition and localization system

quence. Leveraging its recurrent structure, the LSTM can progressively integrate spatial neighborhood information across layers[29]. Given an input sequence x_i, the encoder generates a sequence representation by recursively computing the hidden states, as shown in Equation (2):

$$h_t^{enc} = \mathrm{LSTM}(x_i, h_{t-1}^{enc}) \tag{2}$$

Where, $\mathrm{LSTM}(\cdot)$ denotes the LSTM network used to update the decoder state; h_t^{enc} denotes the hidden state corresponding to the t-th coordinate, which captures the contextual information of all picking points from the beginning up to the current step. After

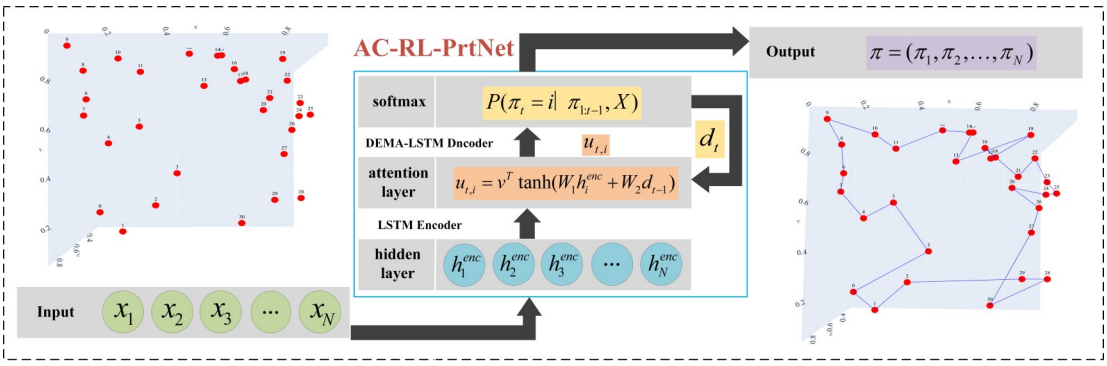


Fig. 3 The process of modelling input to output of AC-RL-PrtNet

processing by the encoder, all hidden states are aggregated into a matrix, which serves as the input for the subsequent attention mechanism and decoding process. The initial hidden state h_0^{enc} was set to a zero vector. To determine an appropriate hidden layer dimension, an ablation study was conducted with three configurations: 64, 128 and 256. The models were evaluated based on total training time, best path length achieved, and model size. As shown in Table 4, the configuration with 128 hidden units achieved the best trade-off between optimization quality and computational cost. Therefore, a hidden size of 128 was adopted in the final model design.

Table 4 Ablation results for hidden state dimension

Hidden dim	Training time/s	Best path	Model size/MB
64	1 672	11.856	4.251
128	1 998	11.414	4.462
256	2 920	11.186	5.282

2) Decoder: The decoder adopts an attention-driven LSTM structure, which takes as input the output from the previous time step and utilizes the contextual information produced by the encoder to generate an action probability distribution. The initial hidden state of the decoder, $d_0 = h_N^{enc}$, was set to the final hidden state of the encoder. The decoding process at time step t involves the following steps:

(a) Attention weight calculation is shown in Equation (3) :

$$u_{t,i} = v^T \tanh\left(\boldsymbol{W}_1 h_i^{enc} + \boldsymbol{W}_2 d_{t-1}\right) \tag{3}$$

Where, $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{h\times d}$, $v \in \mathbb{R}^d$ are learnable parameters; h_i^{enc} denotes the hidden state of the encoder at position; i and d denotes the hidden layer dimension.

(b) Probability distribution generation is shown in Equation (4):

$$P\left(\pi_t = i \middle| \pi_{1:t-1}, X\right) = \text{softmax}\left(u_t, i\right) \tag{4}$$

Where, $P\left(\pi_t = i|\pi_{1:t-1}, X\right)$ denotes the probability of selecting node i at step t, conditioned on the previously visited sequence$\pi_{1:t-1}$ and the input feature set X.

(c) State update is shown in Equation (5):

$$d_t = LSTM\left(\text{Embedding}\left(\pi_t\right), d_{t-1}\right) \tag{5}$$

Where, d_{t-1} represents the hidden state of the decoder at the previous time step, and $\text{Embedding}(\cdot)$ maps the index of the selected node to an embedding vector. To prevent repeated visits, a masking mechanism is employed during the calculation of $P\left(\pi_t = i\right)$, where the logits corresponding to already visited nodes are assigned a value of $-\infty$, thereby ensuring that only feasible paths are generated.

1.3.2 MDPmodeling

RL enables agents to interact with their environment to learn optimal strategies. In the context of the TSP, the problem solving process can be formulated as a MDP. Specifically, the state space is defined as the set of unvisited safflower coordinates at each step. For example, the state at time step t is defined as: $s_t = \{ x_i \middle| i \notin \pi_{1:t-1} \}$ indicating the set of nodes not yet visited up to step t. Transitions between states correspond to selecting an unvisited node to visit next. For instance, given the current state s_t, executing an action (selecting a new target node) results in the next state: $s_{t+1} = s_t \backslash \{ x_i \}$.

In the reinforcement learning framework, the reward function is defined based on the negative Euclidean distance between consecutive nodes, as shown in Equation (6):

$$r_t = - \| x_{\pi_t} - x_{\pi_{t+1}} \|^2 \tag{6}$$

Where, $\pi_{N+1} = \pi_1$ represents the return to the starting node. The immediate reward r_t denotes the negative squared Euclidean distance between two consecutive nodes. The total return $R(\pi)$ is the negative

sum of travel distances along the path as Equation (7):

$$R(\pi)=\sum_{t=1}^{N} r_t=-L(\pi) \tag{7}$$

Where, $L(\pi)$ denotes the length of the current path under policy π.

To guide the agent's behavior, a neural network based on parameterized stochastic policy function is adopted as Equation (8):

$$\pi_\theta(a_t|s_t)=P(\pi_t=i|\pi_{1:t-1},X) \tag{8}$$

Where, X represents the set of all safflower coordinates; $\pi_\theta(a_t|s_t)$ denotes the stochastic policy parameterized by θ, which gives the probability of selecting action a_t under the current state s_t. The policy outputs a probability distribution over the available actions (unvisited nodes) and determines the likelihood of selecting each one at the current step.

By combining the policy network with a value function estimator, the training is performed using the advantage actor-critic algorithm, which allows for efficient gradient based policy optimization.

1.3.3　Policy gradient method

The policy gradient method optimizes the policy parameters θ directly to maximize the expected cumulative reward. The core idea is to increase the probability of actions leading to high-return trajectories, while suppressing those associated with lower returns. The objective function is defined as the expected return, as shown in Equation (9):

$$J(\theta)=E_{\pi_\theta}\left[R(\pi)\right] \tag{9}$$

Where,$J(\theta)$ denotes the expected cumulative reward under policy π_θ; θ represents the learnable parameters of the policy network.

According to the policy gradient theorem[30], the gradient can be written as shown in Equation (10):

$$\nabla_\theta J(\theta)=E_{\pi_\theta}\left[\sum_{t=1}^{N}\nabla_\theta \log \pi_\theta(a_t|s_t)\cdot Q^{\pi_\theta}(s_t,a_t)\right] \tag{10}$$

Where, $Q^{\pi_\theta}(s_t,a_t)$ denotes the action-value function under policy π_θ, representing the expected return after taking action a_t in state s_t. Since directly computing Q requires trajectory rollout, which is computationally expensive, it is often approximated by the empirical return, as shown in Equation (11):

$$Q^{\pi_\theta}(s_t,a_t)\approx\sum_{k=t}^{N} r_k \tag{11}$$

Thus, the gradient update can be simplified as shown in Equation (12):

$$\nabla_\theta J(\theta)\approx\frac{1}{B}\sum_{i=1}^{B}\sum_{t=1}^{N}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\cdot\left(\sum_{k=t}^{N} r_k^i\right) \tag{12}$$

Where, B is the batch size. However, this estimation often suffers from high variance, which may hinder stable convergence. To mitigate this issue, a baseline function was introduced to reduce variance without biasing the gradient, as expressed in Equation (13):

$$\nabla_\theta J(\theta)=E_{\pi_\theta}\left[\sum_{t=1}^{N}\nabla_\theta \log \pi_\theta(a_t|s_t)\cdot\left(Q^{\pi_\theta}(s_t,a_t)-b(s_t)\right)\right] \tag{13}$$

1.3.4　Actor-critic architecture and policy

To address the complexity of 3D safflower picking, this study adopts a collaborative optimization framework based on the actor-critic architecture. The agent interacts with the environment to continuously improve its policy without requiring accurate labeled data. The overall interaction between the environment, the actor network, and the critic network is illustrated in Fig. 4, which demonstrates the flow of information, decision making, and policy update processes involved in this reinforcement learning framework.

The actor network is implemented using a pointer network, essentially serving as a parameterized policy function responsible for generating the probability distribution over actions for dynamic path planning (as in Equation (4)). In the context of the 3D TSP, the actor leverages the attention mechanism in the decoder (as described in Equation (3)) to capture global spatial features extracted by the encoder. It then combines these features with the current state to produce a prioritized ranking of candidate nodes. Unlike traditional greedy algorithms, the actor selects the next node through probabilistic sampling. For example, during training, a multinomial distribution is used to sample actions, ensuring sufficient exploration of the solution space. The policy gradient signal in reinforcement learning (as defined in Equation (12)) directly influences this probability distribution, increasing the likelihood of actions associated with high-reward trajectories while suppressing the generation of inefficient paths.

The critic network $V_\phi(s)$ functions as an independent evaluation module, responsible for estimating the expected return $E\left[R(\pi)|s_t\right]$ under the current state s_t and policy π_θ. Structurally, it resembles the encoder, but employs a parameter separated LSTM network (nonshared with the actor encoder). By aggregating

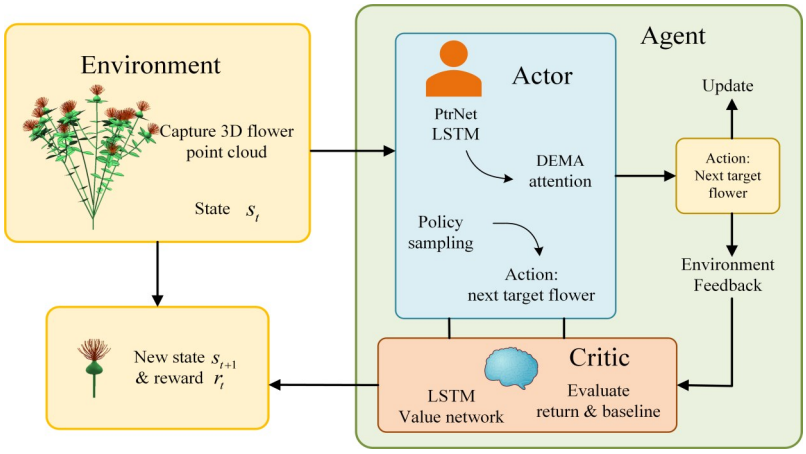


Fig. 4 An interactive actor-critic decision framework for 3D safflower picking tasks

the hidden representations of unvisited nodes (as in Equation (2)), the critic produces a global value representation. A key role of the critic is to provide a dynamic baseline (Equation (3)), which transforms the original Monte Carlo return $R(\pi)$ into the advantage function, as shown in Equation (14):

$$A(s_t, a_t) = R(\pi) - V_\phi(s_t) \tag{14}$$

Where, $A(s_t,a_t)$ denotes the advantage of taking action a_t at state s_t under policy π; $V_\phi(s_t)$ represents the critic's estimate of the expected return from state s_t with parameters $\varnothing$. This formulation effectively reduces the variance of gradient estimation and enhances the stability of policy optimization.

The critic loss function aims to minimize the mean squared error between the predicted value function $V_\phi(s)$ and the actual expected return, thereby driving $V_\phi(s)$ to approximate the expected return more accurately, as defined in Equation (15) :

$$L_{critic} = \frac{1}{B}\sum_{i=1}^{B}\left(R(\pi^i) - V_\phi(s)\right)^2 \tag{15}$$

Where, $R(\pi^i)$ is the total return of the i-th sampled trajectory.

The actor loss function maximizes the advantage weighted log probability (Equation (16)), adjusting the policy optimization direction based on the advantage value:

$$L_{actor} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{t=1}^{N} A(\pi^i)\cdot \log \pi_\theta\left(a_t^i \middle| s_t^i\right) \tag{16}$$

Where, $A(\pi^i)$ denotes the advantage function of the i-th trajectory; the operator *log* refers to the natural logarithm applied to the policy probability, which is used to compute the policy gradient in the actor-critic framework.

2 Algorithm improvement

In the original RL based pointer network for path planning, the attention mechanism adopts a classical singlehead linear transformation structure. However, this design suffers from the following issues:

1) Static weight allocation: All safflower positions share the same attention computation logic, making the model prone to focusing on local optima rather than the global optimal path.

2) Limited representational capacity: A single attention head struggles to simultaneously capture efficient traversal patterns in dense regions and the priority features of edge positioned flowers. When the flower density varies significantly in the central region, the traditional mechanism often produces intersecting paths with repeated coverage rates as high as 15%~20%[31].

3) Slow inference and decision making: The model exhibits latency during path generation.

To address these issues, this chapter replaces the original attention mechanism with a DEMA attention, which enhances the model's ability to capture spatial information. Additionally, pruning techniques are applied to the LSTM network to further improve inference efficiency and overall model performance.

2.1 Dynamic exponential moving average attention mechanism

The introduction of the EMA-based attention mechanism enables the model to balance current observations with historical attention trends, thereby enhancing temporal stability, smoothing attention distributions, and improving robustness in dynamic spatial

environments[32]. As illustrated in Fig. 5 for the improved DEMA, to improve spatial sensitivity, the improved DEMA attention mechanism incorporates distance encoding directly into the computation process. Initially, the input undergoes multi-head partitioning, during which spatial distance information is embedded to enrich feature representation. This is followed by an adaptive reweighting phase driven by dynamic exponential moving average calculations. Subsequently, a nonlinear projection layer refines the aggregated context into the final attention output. Through the integration of present contextual cues, temporal dynamics, and spatial layout, the module is better equipped to model inter-flower relationships and generate more precise path planning outcomes. The key components of this improved attention pipeline are elaborated in the subsequent subsections.

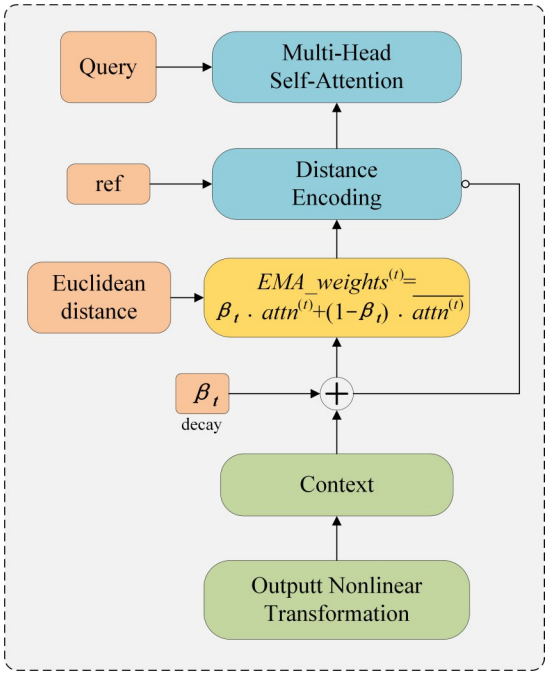


Fig. 5 Computational flow of the DEMA attention mechanism

2.1.1 Redefinition of multi head self attention

The improved attention module divides the embedding dimensions into multiple subspaces, enabling the model to capture diverse environmental features in parallel, such as sharp distributional changes and local density variations. Meanwhile, a dynamic decay parameter is initialized and constrained within a predefined range to balance the influence of the current attention with that of the historical global average attention.

2.1.2 Incorporation of distance encoding

To incorporate spatial topology information into the attention mechanism, a distance-encoding module was introduced. Specifically, the Euclidean distances between flower positions are used as additional inputs, transformed into vector representations through a linear projection, and then fused with the query vectors to compute the attention scores. It is as follows Equation (17):

$$\psi\left(d_{ij}\right) = \boldsymbol{W}_d \cdot d_{ij} + b_d \in R^{d_{head}} \tag{17}$$

Where, d_{ij} denotes the Euclidean distance between the i-th and j-th safflower positions, $\boldsymbol{W}_d$ represents the learnable projection weight matrix, $b_d \in R^{d_{head}}$ is the corresponding bias vector.

This design enables the attention weights to reflect not only semantic similarity but also spatial distance, thereby enhancing the model's ability to capture relative positional relationships between flowers in path planning.

2.1.3 Adaptive EMA weight adjustment

Traditional attention mechanisms typically rely solely on the current state to compute attention scores, without incorporating historical statistical information. To address this, an adaptive EMA weight adjustment strategy is introduced, defined as in Equation (18):

$$EMA\_weights^{(t)} = \beta_t \cdot attn^{(t)} + \left(1 - \beta_t\right) \cdot \overline{attn^{(t)}} \tag{18}$$

where, β_t is a dynamic decay factor, and $attn^{(t)}$ represents the attention scores at the current step. Specifically, $\overline{attn^{(t)}}$ represents the attention scores from the previous time step $t-1$ to make β_t responsive to spatial layout variations. It is defined as a function of local target density, as shown in Equation (19):

$$\beta_t = \frac{1}{1 + \gamma \cdot \rho_t} \tag{19}$$

Where, ρ_t represents the spatial density of picking targets around the current query position at time step$t$, and γ is a tunable scaling parameter. This formulation allows the model to emphasize past attention in high-density regions (smaller β_t) while prioritizing current observations in sparse areas (larger β_t), thereby enhancing stability and adaptability in dynamic environments. This EMA-based adjustment balances the current state with past attention trends, enabling the model to adaptively respond to abrupt environmental changes, maintaining a more stable and globally aware attention distribution. This leads to smoother and more effective path planning.

2.1.4 Nonlinear transformation of output

The EMA-adjusted attention scores are used to compute a weighted sum of the value vectors, producing the attention context, as shown in Equation (20):

$$Context_i = \sum_j EMA\_weights^t \cdot V_j \tag{20}$$

Where, $Context_i$ represents the attention context at time step t; $EMA\_weights^t$ denotes the EMA-adjusted attention weights; V_j refers to the value vectors associated with each input.

The resulting context vector is then passed through a nonlinear transformation using a ReLU activation to enhance feature representation, as shown in Equation (21):

$$\begin{cases} z = \mathrm{ReLU}\left(W_{out1} \cdot Context + b_{out1}\right) \\ Output = W_{out2} \cdot z + b_{out2} \end{cases} \tag{21}$$

The terms W_{out1}, W_{out2} and b_{out1}, b_{out2} denote the weights and biases of the first and second fully connected layers, respectively. This nonlinear transformation further enhances the feature composition capability, enabling the network to better capture complex inter object relationships in the picking task.

2.2 Network pruning

In recent years, the increasing complexity and computational cost of deep neural network models have become major challenges. To reduce the consumption of computational resources without significantly compromising model performance, network pruning techniques have been extensively studied[33, 34]. Among them, structured pruning has attracted considerable attention due to its ability to directly eliminate redundant structures in the computational graph, thereby facilitating hardware acceleration[35-37]. In this work, a structured pruning method is proposed for the gated architecture of LSTM networks, focusing on pruning the input and output gates. Additionally, L1-based pruning is applied to the fully connected layers, aiming to reduce model parameters and accelerate inference.

2.2.1 LSTM gate importance based pruning strategy

Given the structural characteristics of LSTM components in recurrent neural networks, this pruning strategy selectively targets the input and output gates, where parameter redundancy is notably more prevalent. This decision is grounded in the following theoretical considerations:

(1) Parameter distribution analysis: Examination of the gate weight matrices indicates a heavy-tailed distribution, particularly within the input and output gates, suggesting that many weights contribute minimally to the model's expressive capacity.

(2) Criticality of the forget gate: Empirical evidence from initial tests shows that even small perturbations in the forget gate's parameters can lead to a marked drop in performance, implying that its preservation is essential for maintaining temporal stability.

(3) Functional complementarity: The input and output gates are responsible for encoding and exposing temporal information, respectively. Their interdependent functions permit synchronized pruning without compromising the LSTM's representational capability.

2.2.2 Structured pruning implementation for gate structures

For a given weight matrix $\boldsymbol{W}$ (i.e., $\boldsymbol{W}_{ih}$ or $\boldsymbol{W}_{hh}$) and the selected gate indices $g \subset \{1,\cdots,4h\}$, the pruning procedure is carried out as follows. First, determine the range of gate indices for the input gate $g_{in} = \{1, \cdots, h\}$ and the output gate $g_{out} = \{3h + 1, \cdots, 4h\}$.

Next, the pruning threshold is computed. For the selected gate, let $\boldsymbol{W}_g$ denote the corresponding weight parameter vector, and define the pruning ratio. The number of parameters to be pruned is given by Equation (22):

$$k = \left\lfloor \alpha \cdot \left| \boldsymbol{W}_g \right| \right\rfloor \tag{22}$$

Where, k is the number of pruned parameters, and α represents the pruning ratio that determines the overall sparsity level.

The pruning threshold τ is computed as the maximum among the smallest k absolute weight values, as shown in Equation (23):

$$\tau = topk\left(\left\{|\boldsymbol{w}|: \boldsymbol{w} \in \boldsymbol{W}_g\right\}, k, \text{largest} = False\right)[-1] \tag{23}$$

Based on τ a pruning mask M is generated as follows, as shown in Equation (24):

$$M_i = \begin{cases} 0, & \text{if } |w_i| \leqslant \tau, \\ 1, & \text{otherwise.} \end{cases} \tag{24}$$

Only the parameters within the target gate indices are pruned, while weights outside this scope remain unchanged.

Finally, the weight matrix is updated by applying the pruning mask, so that the pruned weight matrix W' is expressed as:

$$\boldsymbol{W}' = \boldsymbol{W}_{\text{before}} \oplus (\boldsymbol{w}_g \odot \boldsymbol{M}) \oplus \boldsymbol{W}_{\text{after}} \tag{25}$$

Where, $\oplus$ denotes the concatenation of the unpruned segments; $\odot$ denotes element wise multiplication.

The pruning operation is applied separately to the input and output gates for both $\boldsymbol{W}_{ih}$ and $\boldsymbol{W}_{hh}$. Experimental validation shows that the encoder adopts a pruning ratio of 35% for both its input and output gates, while the decoder uses a pruning ratio of 25% for these gates.

2.3 Structural enhancements of the critic network

To further improve the value estimation capability for planned paths, this work introduced structural-level enhancements within the critic network. Although these enhancements do not directly impact the decoder's decision-making process, they play a vital role in supplying more stable, structure-aware prediction signals. This indirectly boosts the learning efficiency of the actor module and enhances the overall synergy of the actor-critic framework.

2.3.1 Enhanced input embedding with normalization mechanisms

To increase the expressive power of the original spatial coordinates in high-dimensional latent space, the critic integrates a normalization-augmented embedding layer at the input stage. Specifically, Batch Normalization is applied to raw inputs, followed by two fully connected layers with ReLU activations. This is further regularized by Layer Normalization and Dropout to improve discriminative power and generalization under spatial variability. The embedding process is formulated as follows, as shown in Equation (26):

$$\begin{cases} \tilde{x}_i = \text{ReLU}(W_1 x_i + b_1) \\ e_i = \text{LayerNorm}(W_2 \tilde{x}_i + b_2) \end{cases} \tag{26}$$

Where, $x_i \in \mathbb{R}^3$ represents the raw coordinate of the $i$node; $e_i \in \mathbb{R}^d$ is the embedded feature; W_1, W_2 are trainable weights. Layer normalization and dropout jointly stabilize the feature distribution and mitigate overfitting.

2.3.2 Contextual aggregation from LSTM encoder

The critic network adopts the same LSTM encoder as the actor to generate a sequence of hidden representations for the path, as formulated in Equation (27):

$$H = [h_1, h_2, \cdots, h_n] = \text{LSTM}(e_1, e_2, \cdots, e_n) \tag{27}$$

Where $H \in \mathbb{R}^{B \times n \times d}$encodes the entire sequence and $h_{final} = h_n$ denotes the final hidden state. This serves as a condensed global representation of the path, which is then used as the query in the subsequent attention-based value refinement module.

2.3.3 Lightweight attention-based context fusion

To enhance the critic's sensitivity to critical waypoints along the path, a lightweight attention fusion mechanism is introduced to amplify the response to important spatial cues. This module learns to attend selectively to key positions within the LSTM-encoded trajectory. First, the sequence output H is projected using a 1D convolutional layer to produce reference features, as shown in Equation (28):

$$U_{\text{ref}} = \boldsymbol{W}_{\text{ref}} \cdot H^{\text{T}} \tag{28}$$

Where, $\boldsymbol{W}_{\text{ref}}$ represents the learnable convolutional weight matrix that projects the hidden representations into the reference feature space, while H^{T} is the transposed sequence output obtained from the shared LSTM encoder.

Simultaneously, the query vector h_{final} is mapped to the same dimensionality, as shown in Equation (29):

$$U_q = \boldsymbol{W}_q h_{\text{final}} \tag{29}$$

Where, U_q represents the transformed query feature that aligns with the reference feature space, enabling compatibility computation with U_{ref}, $\boldsymbol{W}_q$ denotes the learnable projection matrix used to map the final hidden state into the query space.

A joint compatibility function is then computed via, as expressed in Equation (30):

$$U = \tanh(U_q + U_{\text{ref}}) \tag{30}$$

Attention weights are derived using a learnable vector $V \in \mathbb{R}^{1 \times d}$, as defined in Equation (31):

$$\alpha = \text{softmax}(V \cdot U) \tag{31}$$

These weights are used to obtain a context vector c via weighted aggregation, as described in Equation (32):

$$c = \sum_{i=1}^{n} \alpha_i h_i \tag{32}$$

Where, c denotes the context vector obtained through weighted aggregation of all hidden representations. The final output for value estimation is obtained by combining the context vector with the original query and feeding it into a prediction MLP, as shown in Equation (33):

$$\hat{L} = \text{MLP}(\text{LayerNorm}(c + h_{\text{final}})) \tag{33}$$

This design allows the critic to capture both the

global structure and local saliency of the path, improving the accuracy of cost prediction and facilitating better guidance for policy updates in the actor network.

2.4 Algorithm learning method

In light of the aforementioned modifications, the improved AC-RL-PtrNet model was employed. Each time a path sequence is generated, the DEMA attention mechanism automatically computes the attention scores, thereby yielding more reasonable picking decisions. The pruning operations also improve the training speed while maintaining model performance with minimal loss. Algorithm 1 outlines the pseudocode for the training process.

In the training task, n is set to 40. The experiments were conducted on a Windows 1 064-bit machine equipped with an NVIDIA GeForce RTX 4060 Ti GPU, an Intel Core i5-14600KF CPU @ 3.5 GHz, and 32 GB of RAM. The implementation utilizes PyTorch 2.0 with CUDA 12.1 under Python 3.12.7. Training samples are randomly generated from $[0.1]\times[0.1]\times[0.1]$ space. To ensure good generalization performance, the model was trained for 2 000 epochs, with each epoch comprising 10 steps. The batch size B was set to 256 and the initial learning rate was 0.001, with a decay factor of 0.96 applied every 5 000 steps. Both the actor and critic networks were trained via the Adam optimizer to minimize their respective loss functions. To clearly describe the optimization process of the proposed model, the detailed training steps of the actor-critic framework were presented in the below algorithm.

Training Algorithm

Input: Number of training iterations S, batch size B

Output: Optimal path a^*
Initialize: π_θ (with internal dynamic EMA attention module), V_ϕ, Optimizer θ and ϕ learning rate η

for t= 1 to S **do:**

The actor network π_θ generates B trajectories via a DEMA-based attention mechanism:
$\{a^1,a^2,\cdots,a^B\}\sim\pi_\theta(a|s)$

Compute reward R(π) and log-probability:
$\log\pi_\theta(a^i|s)$

Value estimation using the critic network:
$V_\phi(s)\leftarrow \mathrm{Critic}(s)$

Compute advantage:
$A^i = R(a^i) - V_\phi(s)$

Update actor: Backpropagate the error through the LSTM-based DEMA module in the actor network:
$\nabla_\theta \leftarrow \mathbb{E}_\pi\left[A(\pi^i)\cdot\left(-\log\pi_\theta(a^i|s)\right)\right]$
$\theta\leftarrow\theta+\eta\cdot\nabla_\theta$

Update critic using mean-squared error loss:
$\nabla_\phi \leftarrow \mathbb{E}_\pi\left[\left(R(\pi^i)-V_\phi(s)\right)^2\right]$
$\phi\leftarrow\phi-\eta\cdot\nabla_\phi$

end for: Return to the shortest path a^*

3 Experiments and results

To validate the effectiveness of the proposed model in addressing the safflower picking sequence planning problem, a simulation-based analysis was conducted. The performance of the model was evaluated using metrics such as path length and average runtime. In this chapter, all safflowers coordinates were normalized to enhance numerical stability and mitigate gradient explosion. In section 3.5, the actual three-dimensional picking paths were reconstructed to provide a visual representation of the real trajectories.

3.1 Impact of DEMA on the model

To evaluate the influence of the DEMA attention mechanism on model performance, analyses were conducted during both the training and testing phases. The baseline model refers to the original unmodified network. In the training phase, comparative experiments were carried out with the baseline model, graph attention network (GAT), efficient multi-scale attention (EMSA), seer attention mechanism (SeA), DEMA without distance information (NoDist-DEMA), and the DEMA-enhanced model. Fig. 6 illustrates the convergence curves for these models during training. The results indicate that the DEMA-based model achieves superior improvements in terms of path length compared to the other methods. Moreover, it demonstrates faster convergence at approximately 200 epochs and avoids getting trapped in local optima, thereby attaining a better convergence value. It is noted, however, that increasing the number of training epochs necessitates more training time.

Based on the inherent randomness in the number of safflower picks per run, the model was tested with n =20, 28, 37, and 46 (as shown in Table 5). The results indicate that, for n=20 and n=37, the improved SeA attention mechanism yields path lengths comparable to those of the DEMA-enhanced model. However, in terms of average runtime, the SeA mechanism requires 1.5 to 2 times longer than the DEMA model. In com-

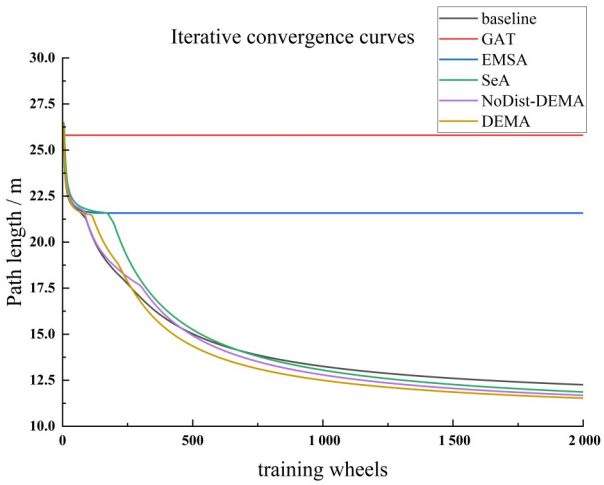


Fig. 6 Iterative convergence comparison of various attention mechanisms against the baseline

parison to the baseline model, the proposed approach not only significantly reduces the path length but also nearly doubles the average inference speed. Compared to the NoDist-DEMA variant, which excludes distance encoding, the full DEMA model consistently achieves marginally shorter path lengths and faster inference across all test cases. This demonstrates that integrating distance information enhances both spatial perception and planning efficiency, further validating the design of the DEMA attention mechanism.

As shown in Table 5, the numerical results have demonstrated the performance differences of various attention mechanisms in terms of path length and computation time across multiple test points. To further visualize these differences and highlight the advantages of the proposed method, Fig. 7 presents a color matrix that fully reflects the differences in path planning quality and robustness among various attention mechanisms at different test points. The numerical values in the figure represent the differences in path performance relative to the baseline model, with larger positive values indicating greater improvements. In terms of path length, the DEMA model demonstrates a more substantial enhancement, especially when the number of safflower picking points ranges from 37 to 50, yielding improvements between 0.20 and 0.41 cm. Moreover, the DEMA model consistently maintains deeper color values across most test points, indicating its superior performance in path quality. Although the SeA model exhibits performance comparable to the DEMA model in certain scenarios, Table 5 shows that its runtime is inferior to that of the DEMA approach. These results indicate that the incorporation of the DEMA attention mechanism can significantly enhance both the planning efficiency and the accuracy of safflower picking.

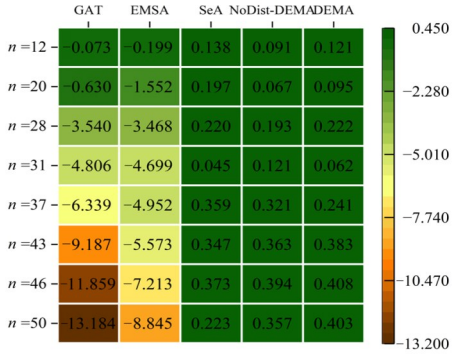


Fig. 7 Path length gap of different models at different test points

3.2 Impact of pruning on the model

Different pruning strategies based on the importance of the gates were designed and evaluated during both the training and testing phases. In this study, three pruning strategies were compared:

Group A: Both the encoder and decoder input and

Table 5 Comparison of results of different architectures under various numbers of safflower points

Method	*n*=20		*n*=28		*n*=37		*n*=46	
	Length/cm	Time/s	Length/cm	Time/s	Length/cm	Time/s	Length/cm	Time/s
baseline	6.769	90.83	8.524	183.26	10.299	368.36	11.806	511.05
GAT	7.399	98.46	12.064	197.85	16.638	414.79	23.665	602.34
EMSA	8.321	126.12	11.992	254.96	15.251	496.07	19.019	582.41
SeA	6.675	63.07	8.404	185.84	9.940	327.68	11.433	510.73
NoDist-DEMA	6.719	59.32	8.346	123.94	10.145	231.76	11.402	304.78
DEMA	6.674	51.01	8.302	113.74	10.058	224.29	11.398	334.62

Note: Length and time values are presented for four different sample sizes (n = 20, 28, 37, 46); Time refers to the duration of sequence planning performed by the model, which is measured in seconds, and length in centimeters.

output gates were pruned by 35%.

Group B: Both the encoder and decoder input and output gates were pruned by 25%.

Group C: The encoder input and output gates were pruned by 35%, while the decoder gates were pruned by 25%.

Table 6 presents the comparative results for these three strategies. Preliminary sensitivity experiments indicated that when the pruning ratio was lower than 25%, the computational benefits were negligible, and when it exceeded 35%, the model experienced severe information loss and degraded performance. Therefore, these three representative configurations were selected for analysis.

Table 6 Comparison of results for different pruning strategies of the proposed model

Strategy	Encoder pruning		Decoder pruning		Params		Sparsity/%	Training time/s	Speedup/%
	Input gate	Output gate	Input gate	Output gate	Initial ($\times10^5$)	Modified ($\times10^5$)			
Group A	0.35	0.35	0.35	0.35		4.106	8	161	—
Group B	0.25	0.25	0.25	0.25	4.462	4.265	4.44	1 747	12
Group C	0.35	0.35	0.25	0.25		4.201	5.88	1 698	15

Note: Speedup (%) represents the relative reduction in training time compared to the unpruned baseline, measured on the same hardware setup. For Group A, "—" means excessive pruning resulted in high sparsity and loss of critical information, causing early convergence during training; thus, the speedup value is not reported.

Analysis shows that the model under Group A achieves an offline training duration of less than 3 minutes; however, the excessive pruning applied to the decoder in Group A leads to significant information loss, causing the model to become trapped in local optima. In contrast, Group C adopted in this study yields a parameter sparsity of approximately 5.88%, effectively removing redundant information and enhancing training speed by about 15%, while simultaneously reducing model size and inference overhead. Therefore, implementing pruning operations on the LSTM not only optimizes computational speed but also decreases memory usage.

3.3 Ablation study

To further evaluate the individual and cumulative contributions of each proposed enhancement, including the DEMA attention mechanism, the structured pruning strategy, and the improved critic network, an ablation study was conducted. Specifically, four configurations were compared: 1) the baseline model without any enhancement, 2) the model with DEMA only, 3) the model with DEMA and pruning, and 4) the full model integrating DEMA, pruning, and the enhanced critic. To this end, we validated this ablation experiment on test sets with 25, 31, and 43 safflowers.

As shown in Table 7, although the full model exhibits slightly longer inference time compared to the previous variant under the 25-target scenario, it consistently achieves superior performance across other datasets in terms of path length reduction and inference robustness. These results indicate that the proposed model enhancements are well-suited for sequential path planning in safflower picking tasks.

Table 7 Ablation study on model enhancements

baseline	DEMA	Pruning	The enhanced critic	*n*=25		*n*=31		*n*=43	
				Length/cm	Time/s	Length/cm	Time/s	Length/cm	Time/s
√	×	×	×	6.674	88.08	6.749	167.35	11.426	497.50
×	√	×	×	5.930	80.86	6.647	127.00	11.049	262.47
×	√	√	×	5.907	63.00	6.441	125.00	11.043	244.27
×	√	√	√	5.826	69.79	6.356	94.90	10.784	244.50

Note: √ indicates that the corresponding component is included in the model; × indicates that the component is not used.

3.4 Comparison with swarm intelligence algorithms

The performance of the proposed model (before and after improvement) was compared with traditional swarm intelligence algorithms, including PSO, the non-dominated sorting genetic algorithm (NSGA), and ACO, in the task of safflower picking path planning.

The evaluation was conducted using two test cases with safflower quantities of n=25 and n=31, focusing on metrics such as path length, computational time, and speed improvement.

As shown in Table 8, swarm intelligence algorithms tend to require significantly longer runtimes in real-world deployment scenarios. Using AC-RL-PtrNet as the baseline for improvement rate comparisons, test cases with n = 25 and n = 31 safflowers were evaluated. The proposed model achieved inference speed improvements ranging from −2.63% to 61.87% for the 25-safflower case and 22.93% to 59.10% for the 31-safflower case, along with path length reductions between 4.24% and 61.47%, compared to traditional swarm intelligence algorithms such as PSO, ACO, and NSGA. Improvement rate (%) represents the relative time-saving of AC-RL-PtrNet compared to each method, calculated as $(T_{method} - T_{AC\text{-}RL})/T_{method} \times 100\%$. A higher value indicates that AC-RL-PtrNet provides a more significant improvement over the other methods. To assess the statistical significance of these improvements, independent two-sample t-tests were performed on 30 repeated runs. The results in Table 8 indicate that this method achieves statistically significant path length reductions over all baselines ($p < 0.05$), confirming the robustness of the observed gains.

Table 8 Comparison of the proposed AC-RL-PtrNet model with traditional algorithms

Method	n=25				n=31			
	Length/cm (p-value)	Improvement rate/%	Time/s	Improvement rate/%	Length/cm (p-value)	Improvement rate/%	Time/s	Improvement rate/%
Baseline	6.084 (p<0.008 0)	4.24	82.00	14.85	6.913 (p<0.005 0)	8.07	215.00	55.86
PSO	15.123 (p<0.001 0)	61.47	183.00	61.87	15.438 (p<0.001 0)	58.82	232.00	59.10
ACO	7.090 (p<0.001 0)	17.84	104.00	32.91	6.966 (p<0.001 0)	8.75	123.00	22.93
NSGA	6.440 (p<0.003 6)	9.56	68.00	−2.63	7.960 (p<0.001 0)	20.17	135.00	29.70
AC-RL-PtrNet	5.826	—	69.79	—	6.356	—	94.90	—

Note: P-values represent the statistical significance of path length differences between each method and the proposed AC-RL-PtrNet. A smaller p-value (typically < 0.05) indicates that the observed difference is unlikely to have occurred by chance, thus validating the performance gain; "—" means the comparison of AC-RL-PtrNet with itself, where improvement is not applicable.

In the test case with 25 safflowers, although the NSGA algorithm achieves a shorter runtime than the improved AC-RL-PtrNet model, the latter yields a 9.56% improvement in path optimization, demonstrating a significantly higher planning efficiency. It is important to note that reinforcement learning operates in an offline training and online inference paradigm. This means that as the number of safflower targets increases, this model does not require retraining for each new instance, unlike swarm intelligence algorithms, which need to re-optimize from scratch for different configurations. This advantage makes this approach more suitable for real-time, responsive applications in safflower picking scenarios.

Furthermore, as shown in Table 8, the AC-RL-PtrNet generates shorter paths than swarm intelligence algorithms in both the n=25 and n=31 test cases, with statistically significant improvements. It reduces the path length by 0.258 and 0.557 cm units compared to the baseline, respectively. Notably, the pruning strategy introduced in this model does not result in significant loss of positional information, thereby achieving a desirable balance between path optimization and inference efficiency.

3.5 Field experiment

3.5.1 Field setup and real-world planning examples

To validate the practical feasibility of the proposed method, real-world field experiments were conducted in Jimusaer county, Xinjiang, using this safflower-picking robot in an open agricultural environment. As shown in Fig. 8a, the robot was deployed in a safflower field and executed picking path planning in real time. Two representative test cases involving 25 and 31 safflower blossoms were selected for demonstration, corresponding to different safflower varieties, as illustrated in Fig. 8b and Fig. 8c, respectively. These scenarios provide a practical basis for evaluating the performance of this model under different spatial configurations and picking densities.

a. Field experiment b. 25 safflower scenarios c. 31 safflower scenarios

Fig. 8 Field experiment setup and example planning results for 25 and 31 safflower blossoms

3.5.2 Analysis of sequence planning results

To further evaluate the real-world effectiveness of the proposed model, we conducted a comparative analysis against NSGA, the best-performing baseline among the traditional optimization algorithms, across two representative safflower varieties and picking scenarios—25 and 31 blossoms of Yuhong No.1 and Yunhong No. 5, respectively. The results, summarized in Table 9, demonstrate that the AC-RL-PtrNet model consistently outperformed NSGA in both path and time efficiency. Specifically, this model achieved reductions of 11.52% and 20.17% in path length, as well as 5.43% and 29.70% in picking time across the two scenarios.

Table 9 Comparison of path and time efficiency of AC-RL-PtrNet and NSGA in two fields picking scenarios

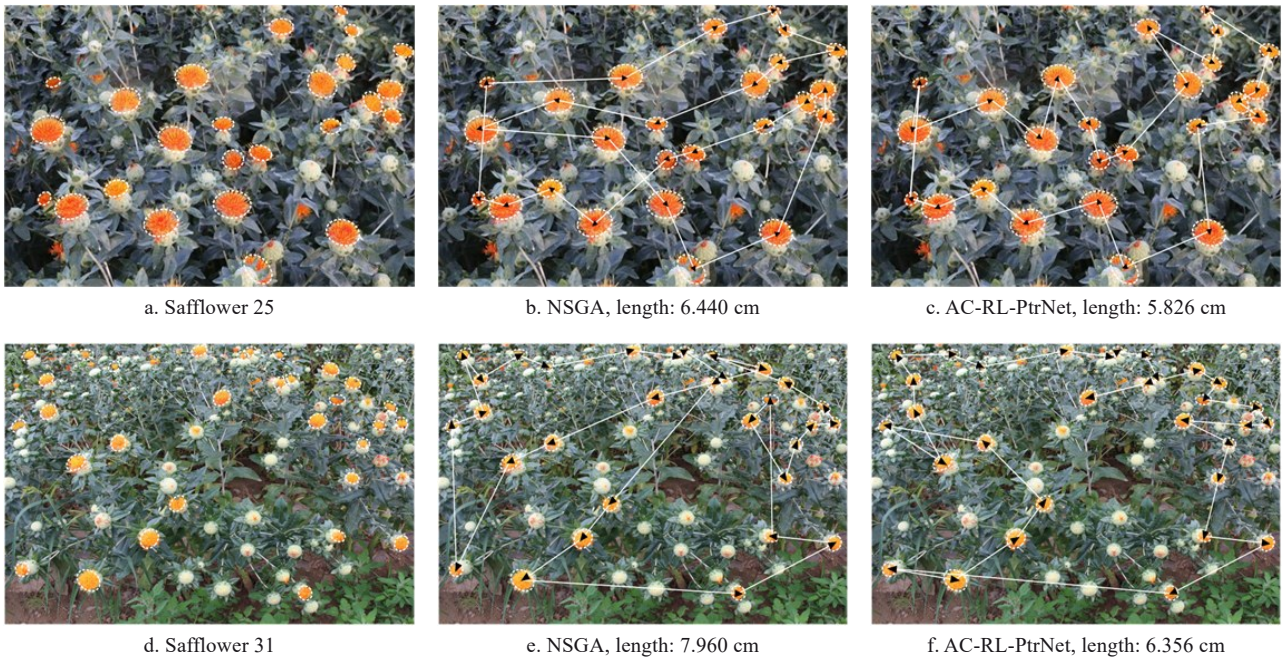
Scenario	Number	Variety	Path saving/%	Time saving/%
25 safflower scenario	25	Yuhong No.1	11.52	5.43
31 safflower scenario	31	Yunhong No.5	20.17	29.70

The corresponding sequence planning visualizations are presented in Fig. 9. Fig. 9a and Fig. 9d correspond to the sequence planning results for 25 and 31 safflowers, respectively. Figures 9b, 9c, 9e, 9f illustrate the comparison between NSGA and AC-RL-PtrNet under the two scenarios. The visual comparison highlights that the paths generated by the AC-RL-PtrNet model are not only shorter but also more structured and efficient than those produced by the NSGA algorithm, which tends to generate redundant and crossover-prone paths. In contrast, the AC-RL-PtrNet model demonstrates superior balance between global path optimization and local adjustment, producing more regular and structured paths. The node connections are more consistent with the spatial distribution of the targets, revealing a more ideal 3D TSP planning capability, as well as enhanced intelligence and practical applicability. These visual results further confirm that the proposed approach can effectively generate optimal picking sequences even in complex environments characterized by densely clustered or unevenly distributed safflowers.

In summary, the improved model exhibits notable advantages in extracting spatial distribution features of safflowers, while the introduced pruning strategy effectively reduces computational redundancy and enhances decision-making efficiency. These strengths make it a robust, stable, and efficient solution for intelligent safflower picking tasks.

4 Conclusion

This study focuses on the task of automatic safflower picking and, by considering the structural characteristics of a parallel picking robot platform, proposes a sequence planning algorithm based on a reinforcement learning pointer network under the actor-critic framework. The proposed method is specifically designed for integration into safflower-picking robots, addressing the complex path planning needs encountered in unstructured agricultural environments. To address redundant paths and low time efficiency in traditional methods, a DEMA attention mechanism with distance encoding was integrated into the actor network, enhancing its ability to model spatial relationships among safflowers. A structured pruning strategy was also applied to the LSTM gating units and fully connected layers, effectively reducing model complexity while preserving performance. Furthermore, a novel enhancement to the critic network was introduced. This involved a refined embedding and attention formulation, incorporating input normalization, multi-layer residual connections, and an improved prediction module, significantly improving value estimation accu-



a. Safflower 25 b. NSGA, length: 6.440 cm c. AC-RL-PtrNet, length: 5.826 cm

d. Safflower 31 e. NSGA, length: 7.960 cm f. AC-RL-PtrNet, length: 6.356 cm

Fig. 9 Comparison of safflower picking sequence planning on 25 and 31 flowers

racy and training stability. The complementary design between actor and critic components ensures more consistent policy optimization during learning. In the experimental section, the sequential safflower picking problem was reformulated as a three-dimensional traveling salesman problem. Ablation studies reveal that each of the three enhancements, DEMA attention, LSTM pruning, and critic refinement, contributes incrementally to overall performance gains in both inference time and path quality. Furthermore, empirical comparisons with established swarm intelligence algorithm, such as PSO, NSGA and ACO, demonstrated that this method consistently delivers shorter traversal routes and improved planning efficiency under different target distributions. In real-world validation involving datasets of 25 and 31 safflowers, the proposed approach surpasses the strongest baseline NSGA, confirming its practical advantages. These results highlight the framework's applicability in real-time field deployment and its effectiveness in improving path planning capabilities of safflower-picking robots. Additionally, due to the generality of the 3D TSP formulation and the modular design of the AC-RL-PtrNet architecture, this method has strong potential for transferability to other agricultural applications such as tomato, strawberry, or pear picking. The DEMA attention and pruning strategies can also be reconfigured to accommodate different spatial complexities, object densities, and environmental conditions, making the proposed framework a promising solution for a wider class of 3D sequence planning tasks in precision agriculture. These findings affirm the robustness and adaptability of the AC-RL-PtrNet framework in complex agricultural environments, emphasizing its potential for enhancing automated safflower picking performance.

Conflict of interest statement: All authors declare no competing interests.

References:

[1] LU J X, ZHANG C X, HU Y, et al. Application of multiple chemical and biological approaches for quality assessment of *Carthamus tinctorius* L. (safflower) by determining both the primary and secondary metabolites[J]. Phytomedicine, 2019, 58: 152826.

[2] WANG L, CHEN Z, HAN B, et al. Comprehensive analysis of volatile compounds in cold-pressed safflower seed oil from Xinjiang, China[J]. Food Science & Nutrition, 2020, 8(2): 903-914.

[3] MA B J, XIA H, GE Y, et al. A method for identifying picking points in safflower point clouds based on an improved PointNet++ network[J]. Agronomy, 2025, 15(5): 1125.

[4] WANG X R, ZHOU J P, XU Y, et al. Location of safflower filaments picking points in complex environment based on improved YOLOv5 algorithm[J]. Computers and Electronics in Agriculture, 2024, 227: 109463.

[5] ZHANG Z G, WANG Y Z, XU P, et al. WED-YOLO: A detection model for safflower under complex unstructured environment[J]. Agriculture, 2025, 15(2): 205.

[6] LI Y J, FENG Q C, LIU C, et al. MTA-YOLACT: Multitask-aware network on fruit bunch identification for cherry tomato ro-

botic harvesting[J]. European Journal of Agronomy, 2023, 146: 126812.

[7] JIANG Y W, CHEN J, WANG Z W, et al. Research progress and trend analysis of picking technology for Korla fragrant pear[J]. Horticulturae, 2025, 11(1): 90.

[8] ZHAO C J, FAN B B, LI J, et al. Agricultural robots: Technology progress, challenges and trends[J]. Smart Agriculture, 2023, 5(4): 1-15.

[9] DONG Z, ZHANG X H, YANG W J, et al. Ant colony optimization-based method for energy-efficient cutting trajectory planning in axial robotic roadheader[J]. Applied Soft Computing, 2024, 163: 111965.

[10] CHEN D W, IMDAHL C, LAI D, et al. The Dynamic Traveling Salesman Problem with Time-Dependent and Stochastic travel times: A deep reinforcement learning approach[J]. Transportation Research Part C: Emerging Technologies, 2025, 172: 105022.

[11] GUO Z W, FU H, WU J H, et al. Dynamic task planning for multi-arm apple-harvesting robots using LSTM-PPO reinforcement learning algorithm[J]. Agriculture, 2025, 15(6): 588.

[12] LIU C J, ZHONG Y L, WU R L, et al. Deep reinforcement learning based 3D-trajectory design and task offloading in UAV-enabled MEC system[J]. IEEE Transactions on Vehicular Technology, 2025, 74(2): 3185-3195.

[13] JATI G K, KUWANTO G, HASHMI T, et al. Discrete Komodo algorithm for traveling salesman problem[J]. Applied Soft Computing, 2023, 139: 110219.

[14] SOITINAHO R, VÄYRYNEN V, OKSANEN T. Heuristic cooperative coverage path planning for multiple autonomous agricultural field machines performing sequentially dependent tasks of different working widths and turn characteristics[J]. Biosystems Engineering, 2024, 242: 16-28.

[15] UTAMIMA A, REINERS T. Navigating route planning for multiple vehicles in multifield agriculture with a fast hybrid algorithm [J]. Computers and Electronics in Agriculture, 2023, 212: 108021.

[16] GAO R L, ZHOU Q J, CAO S X, et al. Apple-picking robot picking path planning algorithm based on improved PSO[J]. Electronics, 2023, 12(8): 1832.

[17] FANG S P, RU Y, HU C M, et al. Planning of takeoff/landing site location, dispatch route, and spraying route for a pesticide application helicopter[J]. European Journal of Agronomy, 2023, 146: 126814.

[18] LI X M, GENG L B, LIU K Z, et al. Model-based offline reinforcement learning for AUV path-following under unknown ocean currents with limited data[J]. Drones, 2025, 9(3): 201.

[19] SHEN J, CHEN M C, ZHANG Z C, et al. Model-based offline policy optimization with distribution correcting regularization [C]// Machine Learning and Knowledge Discovery in Databases. Research Track. Cham, Germany: Springer, 2021: 174-189.

[20] SHARMA G, SINGH A, JAIN S. DeepEvap: Deep reinforcement learning based ensemble approach for estimating reference evapotranspiration[J]. Applied Soft Computing, 2022, 125: 109113.

[21] ZHANG Q, FANG X W, GAO X D, et al. Optimising maize threshing process with temporal proximity soft actor-critic deep reinforcement learning algorithm[J]. Biosystems Engineering, 2024, 248: 229-239.

[22] YANG J C, NI J F, LI Y, et al. The intelligent path planning system of agricultural robot *via* reinforcement learning[J]. Sensors, 2022, 22(12): 4316.

[23] LIN G C, ZHU L X, LI J H, et al. Collision-free path planning for a guava-harvesting robot based on recurrent deep reinforcement learning[J]. Computers and Electronics in Agriculture, 2021, 188: 106350.

[24] SANTIYUDA G, WARDOYO R, PULUNGAN R. Solving biobjective traveling thief problems with multiobjective reinforcement learning[J]. Applied Soft Computing, 2024, 161: 111751.

[25] BELLO I, PHAM H, LE Q V, et al. Neural combinatorial optimization with reinforcement learning[EB/OL]. arXiv: 1611.09940, 2016.

[26] GU S S, YANG Y. A deep learning algorithm for the max-cut problem based on pointer network structure with supervised learning and reinforcement learning strategies[J]. Mathematics, 2020, 8 (2): 298.

[27] LIN G C, XIONG J T, ZHAO R M, et al. Efficient detection and picking sequence planning of tea buds in a high-density canopy [J]. Computers and Electronics in Agriculture, 2023, 213: 108213.

[28] WANG X R, XU Y, ZHOU J P, et al. Safflower picking recognition in complex environments based on an improved YOLOv7[J]. Transactions of the Chinese Society of Agricultural Engineering, 2023, 39(6): 169-176.

[29] BELHADI A, DJENOURI Y, BELBACHIR A N, et al. Shapley visual transformers for image-to-text generation[J]. Applied Soft Computing, 2024, 166: 112205.

[30] SUTTON R S, MCALLESTER D, SINGH S, et al. Policy gradient methods for reinforcement learning with function approximation[C]// Proceedings of the 13th International Conference on Neural Information Processing Systems. New York, USA: ACM, 1999: 1057-1063.

[31] BRAUWERS G, FRASINCAR F. A general survey on attention mechanisms in deep learning[J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(4): 3279-3298.

[32] LIU Y, ZHANG C, HANG B, et al. An audio attention computational model based on information entropy of two channels and exponential moving average[J]. Human-Centric Computing and Information Sciences, 2019, 9(1): 7.

[33] HE Y, XIAO L G. Structured pruning for deep convolutional neural networks: A survey[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024, 46(5): 2900-2919.

[34] WANG Z, LI C C, WANG X Y. Convolutional neural network pruning with structural redundancy reduction[C]// 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway, New Jersey, USA: IEEE, 2021: 14908-14917.

[35] JIANG P C, XUE Y, NERI F. Convolutional neural network pruning based on multi-objective feature map selection for image classification[J]. Applied Soft Computing, 2023, 139: 110229.

[36] CHEN J L, ZHAO P, CAO X L, et al. Lightweight YOLOv8s-based strawberry plug seedling grading detection and localization *via* channel pruning[J]. Smart Agriculture, 2024, 6(6): 132-143.

[37] ZHU K H, HU F Y, DING Y B, et al. A comprehensive review of network pruning based on pruning granularity and pruning time perspectives[J]. Neurocomputing, 2025, 626: 129382.

DEMA-3D TSP：一种应用在红花采摘机器人序列规划的融合DEMA注意力机制强化学习算法

李萌浩[1]，王小荣[1,2,3*]，刘子贺[1]，段孟渝[1]，金正阳[1]

（1. 新疆大学机械工程学院，新疆乌鲁木齐 830017，中国；
2. 新疆维吾尔自治区农牧机器人及智能装备工程研究中心，新疆乌鲁木齐 830017，中国；
3. 新疆大学工程训练中心，新疆乌鲁木齐 830017，中国）

摘　要：［**目的/意义**］红花自动化采摘中存在多个关键难题，尤其是路径规划效率低、路线质量欠佳，以及在动态复杂环境下决策能力受限等问题。为应对上述挑战，将采摘路径规划任务建模为三维旅行商问题，并提出了一种改进的强化学习框架演员-评论家强化学习指针网络（Actor-Critic Reinforcement Learning Pointer Network AC-RL-PtrNet）。该模型专为农业环境下的智能红花采摘机器人设计，可实现高效的路径优化与决策控制。［**方法**］首先，为克服传统注意力机制在复杂空间结构和动态环境中的固有局限性，提出了一种基于动态指数移动平均框架的增强型注意力模块。该模块融合了多头注意力、空间距离编码和自适应指数平滑机制，使模型能够更充分地捕捉红花目标间的长程依赖关系与空间上下文特征。同时，为在保持推理精度的条件下降低计算开销，采用了结构化剪枝方法，对长短期记忆网络的门控单元及全连接层中的冗余连接进行选择性移除。在此基础上，对评论家网络进行了重新设计，引入了批归一化、残差特征聚合，以及多层价值估计头，从而提升学习稳定性与预测精度，并增强了策略训练过程中演员-评论家（Actor-Critic）之间的协同效果。［**结果和讨论**］为定量评估各模块的性能影响，设计并开展了多组消融实验。结果表明，各功能模块均能带来独立的性能提升，而其组合使用则在路径规划精度与推理效率方面取得了最优表现。该协调的Actor-Critic设计有效提升了轨迹质量与决策稳定性，这对于序列式机器人采摘任务尤为关键。与传统群体智能算法相比，所提出的AC-RL-PtrNet模型在25个目标采摘任务中实现了-2.63%～61.87%的规划时间提升，在31个目标任务中提升幅度为22.93%～59.10%。同时，不同规划实例下的路径长度均显著缩短，表明该模型在不同问题规模下具备良好的泛化与稳定性能。田间实测结果进一步验证了其实际应用效果：在真实红花采摘环境中，AC-RL-PtrNet在25个目标任务中实现路径长度减少9.56%、时间缩短5.43%；在31个目标任务中路径减少20.17%、时间节省29.70%。［**结论**］本研究提出的基于强化学习的结构化路径规划方法在路径规划效率提升与路线质量优化方面均展现出显著优势，为红花采摘机器人在复杂动态环境下实现高效与稳健的自主采摘提供了切实可行的技术方案，同时为解决三维组合优化类问题提供了新的研究思路与方法论支持。

关键词：动态指数移动平均机制；结构化剪枝；强化学习；三维旅行商问题；红花采摘；机器人

基金项目：新疆维吾尔自治区青年科学基金项目（2023D01C190）；新一代人工智能国家科技重大专项（2022ZD0115801）

作者简介：李萌浩，硕士研究生，研究方向为农业路径规划。E-mail：LiMenghao@stu.xju.edu.cn
***通信作者：**王小荣，博士，高级工程师，研究方向为精准农业。E-mail：xiaorongwang@xju.edu.cn