




Article

Dynamic Task Planning for Multi-Arm Harvesting Robots Under Multiple Constraints Using Deep Reinforcement Learning

Feng Xie ^{1,2,†} , Zhengwei Guo ^{2,3,†}, Tao Li ^{2,4,*} , Qingchun Feng ² and Chunjiang Zhao ^{3,*} 

¹ School of Agricultural Engineering, Jiangsu University, Zhenjiang 212000, China; 2111916009@stmail.ujs.edu.cn

² Intelligent Equipment Research Center, Beijing Academy of Agriculture and Forestry Sciences, Beijing 100097, China; gzwneau@163.com (Z.G.); fenqqc@nercita.org.cn (Q.F.)

³ School of Mechanical Engineering, Guangxi University, Nanning 530000, China

⁴ Key Laboratory of Modern Agricultural Intelligent Equipment in South China, Ministry of Agriculture and Rural Affairs, Guangzhou 510000, China

* Correspondence: lit@nercita.org.cn (T.L.); zhaocj@nercita.org.cn (C.Z.)

† These authors contributed equally to this work.

Abstract: Global fruit production costs are increasing amid intensified labor shortages, driving heightened interest in robotic harvesting technologies. Although multi-arm coordination in harvesting robots is considered a highly promising solution to this issue, it introduces technical challenges in achieving effective coordination. These challenges include mutual interference among multi-arm mechanical structures, task allocation across multiple arms, and dynamic operating conditions. This imposes higher demands on task coordination for multi-arm harvesting robots, requiring collision-free collaboration, optimization of task sequences, and dynamic re-planning. In this work, we propose a framework that models the task planning problem of multi-arm operation as a Markov game. First, considering multi-arm cooperative movement and picking sequence optimization, we employ a two-agent Markov game framework to model the multi-arm harvesting robot task planning problem. Second, we introduce a self-attention mechanism and a centralized training and execution strategy in the design and training of our deep reinforcement learning (DRL) model, thereby enhancing the model's adaptability in dynamic and uncertain environments and improving decision accuracy. Finally, we conduct extensive numerical simulations in static environments; when the harvesting targets are set to 25 and 50, the execution time is reduced by 10.7% and 3.1%, respectively, compared to traditional methods. Additionally, in dynamic environments, both operational efficiency and robustness are superior to traditional approaches. The results underscore the potential of our approach to revolutionize multi-arm harvesting robotics by providing a more adaptive and efficient task planning solution. We will research improving the positioning accuracy of fruits in the future, which will make it possible to apply this framework to real robots.

Keywords: multi-arm harvesting robots; target planning; multiple constraints; deep reinforcement learning



Academic Editor: Esmaeil Fallahi

Received: 31 October 2024

Revised: 1 January 2025

Accepted: 5 January 2025

Published: 14 January 2025

Citation: Xie, F.; Guo, Z.; Li, T.; Feng, Q.; Zhao, C. Dynamic Task Planning for Multi-Arm Harvesting Robots Under Multiple Constraints Using Deep Reinforcement Learning. *Horticulturae* **2025**, *11*, 88. <https://doi.org/10.3390/horticulturae11010088>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Fruit harvesting is labor-intensive, heavily reliant on seasonal workers, and results in high production costs [1]. The growing labor shortage and rising expenses have made this challenge more acute. Autonomous harvesting by robots has emerged as a promising solution, attracting significant attention for its potential to enhance efficiency, reduce labor dependence, and lower costs. Robots offer a crucial solution to the growing need

to replace human labor in agriculture. For more than three decades, research has focused on robotic harvesters that autonomously detect fruits, plan paths, and execute picking [2]; these were made possible by the advent of computer vision and sensor technology and the automated detection of fruits in the early 1990s; path planning and navigation technologies, with GPS and SLAM technologies helping robots navigate autonomously, in the 2000s; and breakthroughs in robotic arms and intelligent control systems that made harvesting operations more accurate and efficient in the 2010s. In recent years, driven by rapid advancements in disciplines such as sensor technology, materials science, and computer science, harvesting robots have achieved significant breakthroughs in key areas. These include decision-making, perception and localization [3], structural optimization [4], intelligent control [5], and operational control [6], all of which are critical to improving harvesting operations. These advancements led to a wide range of harvesting robots for fruits such as apples [7], citrus [8], and tomatoes [9].

As the commercialization and practical application of harvesting robots accelerate, operational efficiency has become a key performance indicator and is gaining increasing attention. Multi-arm robots have the potential to significantly enhance efficiency, leading to numerous studies and the commercialization of robots like the 4-arm kiwifruit robot [10], the 24-arm strawberry robot [11], and the 12-arm apple robot [12].

According to the number of manipulators that can be driven, there are currently two types of harvesting robots: single-arm robots and multi-arm robots. For crops such as strawberries and tomatoes cultivated in greenhouses, the robot's workspace is relatively confined, leading to the common use of single-arm designs. However, in orchard harvesting, especially for fruits like apples, which are distributed within tall canopies, robots require larger working areas and higher harvesting efficiencies. Multi-arm robots are more frequently employed in such scenarios. In contrast to single-arm robots, the cooperative operation of multi-arm robots is pivotal for efficiency [13]. Cooperative operation refers to the coordination and collaboration between robotic arms, including task allocation, synchronized actions, and the efficient utilization of spatial resources. A parallel collaboration mode has emerged for multi-arm robots like FFR [12], AGROBOT [14], and Harvest CROO [15]. Within this framework, each arm operates independently, working in parallel to reduce the time required per fruit harvest. However, under actual working conditions, the fruits to be harvested may not be evenly distributed across each arm's working area. Instead, the fruits may be sparsely distributed in some areas and occur in clumps in other areas. When designing multi-arm harvesting robots, the working space of individual arms may overlap to cover a broader harvesting area. Additionally, many robotic arms are structurally designed to maintain a minimum distance between arms to prevent physical collisions and interference [16]. Moreover, the coupling between joints restricts the independent operating space of each arm. As a result, in large-scale operations, physical interference, motion constraints, and uneven fruit distribution can result in some arms entering a state of vacancy, meaning that the arm is unable to perform harvesting tasks for a period of time. This results in a decrease in overall harvesting efficiency.

To address the issue of reduced harvesting efficiency caused by some arms entering a state of vacancy in the multi-arm collaboration of harvesting robots, a new cooperative harvesting mode for multi-arm robots is proposed. This mode involves overlapping working zones, allowing arms to be assigned targets based on fruit distribution. Considerations for interference, coverage, and efficiency are crucial in this cooperative approach. Barnett et al. conducted an in-depth study on a multi-arm kiwifruit harvesting robot, focusing on task partitioning and reachability. Their work aimed to ensure uniform fruit distribution across the harvesting area while simultaneously reducing task completion time, which is crucial for enhancing overall operational efficiency in agricultural robotics [17,18]. Building

on this, Mann et al. optimized the task assignment for each manipulator to maximize harvest yield. They conducted a comprehensive analysis of the optimal robot configuration, considering factors such as the number of arms, manipulator capabilities, and operational speed [19]. Despite these advancements, a notable gap remains in the existing literature. The aforementioned studies primarily addressed task allocation and robot configuration, but they did not explore the impact of the picking sequence on operational efficiency. The order in which fruits are harvested can significantly influence overall productivity and resource utilization, making this an important consideration in the design of harvesting strategies. Thus, while Barnett et al. and Mann et al. contribute valuable insights into the optimization of robotic harvesting systems, the limited focus on picking sequences indicates a critical area for further research. Addressing this gap could lead to substantial improvements in the efficiency and effectiveness of autonomous harvesting solutions.

Addressing the picking sequence is crucial, and rationally assigning targets and sequences to each arm can be considered a multi-objective task planning problem [20]. The objective of this problem is to allocate tasks to the manipulators under certain constraints to optimize overall system performance. Such problems are highly challenging in the field of cooperative operations and have garnered extensive attention across diverse sectors, such as workshop scheduling [21], traffic signal control [22], and energy management [23]. Some researchers have defined these problems as optimization problems, aiming to find the optimal solution for an objective function under given constraints [24]. Linear programming [25] is a classic method that transforms actual task allocation problems into mathematical models to apply optimization techniques. However, certain optimization problems cannot be solved by polynomial-time algorithms, and such problems are known as NP-hard problems [26]. Currently, many task planning problems, including multi-arm harvesting, which is the focus of this study, fall under the category of NP-hard problems. Heuristic methods [27] are considered effective means for solving NP-hard problems; they are heuristic stochastic optimization algorithms that can quickly find near-optimal solutions. Swarm intelligence algorithms are particularly widely used in the field of multi-robot task planning, such as in intelligent warehousing [28], vehicle routing [29], multi-robot assembly [30], and UAV planning [31]. Tao et al. applied a Multi-Traveling Salesman Problem to model the task planning of a multi-arm apple harvesting robot. By considering asynchronous overlapping constraints and using a double-chromosome genetic algorithm to calculate picking sequences, they potentially increased harvesting efficiency by up to 4.25 times [32]. However, heuristic algorithms may not be able to effectively respond to future disturbances, such as picking failures.

In real-world orchard harvesting operations, a high success rate for each target pick is not guaranteed. Additionally, rigid collisions between the robot and tree branches, coupled with the inherent uncertainties of outdoor environments, can render certain targets unpickable before the harvesting action is executed. Therefore, task planning for multi-arm harvesting must effectively address picking failures and variations in the positions and quantities of target fruits. These capabilities significantly influence the operational efficiency of multi-arm harvesting robots, underscoring the challenges posed by dynamic environmental changes to the robots. In dynamic environments, robots must use historical information from themselves and other robots to learn how to navigate interruptions and enhance system robustness [33]. In recent years, with the breakthroughs of artificial intelligence, researchers have endeavored to apply AI methods to multi-robot task allocation problems. Reinforcement learning, a quintessential machine learning technique, allows systems to learn how to act in diverse environments based on past information. Wilson et al. applied reinforcement learning methods to multi-task allocation problems, significantly expediting the convergence to the optimal task allocation strategy [34]. Sun et al. inves-

tingated adaptive batching strategies to address environmental adaptation and dynamic batching in task allocation [35]. Choudhury et al. considered the dynamic assignment of tasks to multiple robots under time window constraints and uncertainty of task completion, proposing a multi-robot stochastic conflict model with a deep learning allocation algorithm that improved scalability with increasing robot numbers [36]. It is evident that algorithms based on deep reinforcement learning (DRL) are effective solutions for complex dynamic task planning problems. For multi-arm harvesting robots, modeling the time optimization problem is highly complex due to multiple mechanical constraints, including mechanical interference, joint freedom degree coupling, and multi-arm interaction. Additionally, the need to adjust plans in response to picking failures and newly generated targets highlights the requirement for dynamic planning capabilities. This suggests that heuristic search-based methods are unable to address the aforementioned issues, while deep reinforcement learning methods are not directly applicable; yet, there is a lack of research on task planning for multi-arm harvesting robots.

Accordingly, a framework of dynamic task planning for multi-arm harvesting robots is proposed in this study. These robots are mainly used for harvesting tasks in dwarf and high-density orchards. Due to the uniform planting layout and the relatively narrow canopy of the fruit trees in such orchards, the robots have a high degree of reachability and are very suitable for multi-arm collaborative operations. Moreover, the structural design of the robots has been significantly simplified, which not only improves operational efficiency but also greatly reduces hardware costs. The contributions in our study are as follows:

1. To minimize the overall operational time of the robot, we modeled the problem of task planning as a Markov game and designed the states, actions, state transition, and reward function of the model.
2. To optimize the technical challenges of slow model convergence and insufficient stability, we introduced a self-attention mechanism and a centralized training and execution strategy into the design and training of the deep reinforcement learning (DRL) model. This approach aims to accelerate the model's convergence process and enhance its overall stability.
3. To address the dynamic task planning challenges of multi-arm robots, we propose a task planning framework based on deep reinforcement learning (DRL). Numerical simulation results demonstrate that the framework is both effective and superior.

2. Preliminary Model and Problem Statement

2.1. Robot Model Description

The harvesting robot in this paper comprises four mechanical arms, labeled Arm-1 to Arm-4, as shown in Figure 1. Additionally, this robot comprises subsystems such as a fruit transportation device, an illumination system, and a wheeled mobile platform. Specifically, the robot consists of four robotic arms and grippers, four stereo cameras, a mobile platform, and subsystems of fruit conveyors and illumination. Each robotic arm has a 3-degree-of-freedom Cartesian configuration. Each gripper has three fingers and performs grasping and rotating movements.

The cameras and the computing platform use the USB protocol for communication. The ECMs and the host communicate efficiently via Ethernet for image data transfer, and a switch is used to ensure transmission efficiency. Communication between the host and the servo motors occurs through the CAN bus. Additionally, this paper employs the relay with CAN support and corresponding I/O interfaces to achieve effective control of the host with actuating components such as cylinders, lights, and conveyor belts.

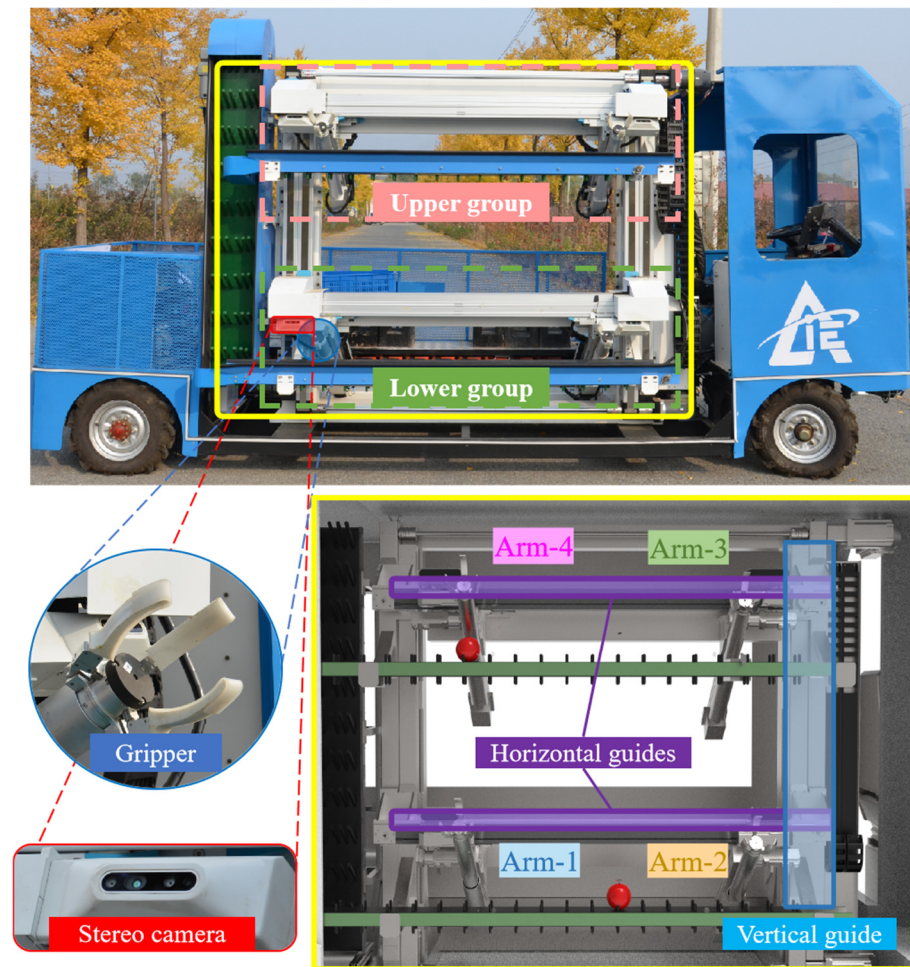


Figure 1. The overview of multi-arm harvesting robots.

Before task planning, each stereo camera captures the color and depth images of the fruits. The color image is processed through a recognition model to determine the position of the target in the image. Combined with the depth image, the three-dimensional coordinates of the fruit are obtained. Then, through coordinate transformation, they are translated from the camera coordinate system to the base coordinate system, yielding all the distribution information about the fruits. This information is used to carry out task planning.

To ensure that each arm covers the entire working space, as shown in Figure 1, Arm-1 and Arm-2 share a horizontal guide, causing their movements along the vertical direction to be coupled. The movements of Arm-3 and Arm-4 are also coordinated, thus dividing the four arms into two groups: upper and lower. Furthermore, these two horizontal guides move on the same vertical guide, providing them access to the middle zone. The aim is to enhance the operational scope of each arm while simultaneously reducing the weight and minimizing hardware and software costs.

It can be seen that these four robotic arms do not operate independently. Due to factors such as joint coupling and mechanical interference, the cooperative operation mechanism of multi-arm harvesting is relatively complex. To express the overall harvest time more efficiently, we first outline the operational behavior of a single robotic arm, as shown in Figure 2. A complete harvest operation for each arm consists of five sequential movements:

1. **Approaching:** The process of moving the horizontal and vertical joint positions from the initial location to the desired one.

2. Extension: The process of moving the extending joint positions from the initial location to the desired one.
3. Grasp: The process of grasping a fruit using a flexible three-finger gripper and rotating the arm.
4. Retraction: The process of moving the extending joint positions from the desired location to the initial one.
5. Placement: The process of releasing a fruit and rotating the arm.

Next, we consider the collaboration of the multi-arm harvesting robot. Due to the coupling of the horizontal guide, only one arm of a group can execute harvesting operations at the same time. This means that the alternation of harvesting is the main form of multi-arm collaboration; an example of this is shown in Figure 2. At first, the four arms were at their ready positions; then, Arm-1 and Arm-4 started to execute their harvest operation (Figure 2a). However, due to the interference of the mechanism, Arm-1 needed to wait until Arm-4 finished the grasping movement (Figure 2b). Arm-1 continued its operation (Figure 2d), which is the alternation of the two groups. To prevent a collision, Arm-4 needed to avoid Arm-1 and allow it to finish its grasping movement first (Figure 2c). Furthermore, the two arms in the same group also perform alternation. In Figure 2e, Arm-1 retracted to place the fruit while Arm-2 approached its target. The coupling of the horizontal guide for Arm-1 and Arm-2 was released when Arm-1 finished the grasping movement. As a result, Arm-2 could access the horizontal guide and perform the approach without waiting for Arm-1 to finish the placement, saving much operating time for the robot.



Figure 2. Cont.



Figure 2. An example of the alternation for multi-arm harvesting robots.

2.2. Picking Time Analysis

The objective of this study is to harvest the maximum number of targets within a limited time frame by optimizing task planning for the robot. The approach is based on the following considerations:

- Additional attempts may be required due to the failure to pick the fruit.
- Mechanical limitations require safety distances in both horizontal and vertical directions. This means that any two joints in the same direction must be at least a certain distance apart.
- The motors driving the joints in the horizontal, vertical, and extension directions rotate at different speeds, resulting in different joint velocities in the three directions.
- The total harvesting time is determined by the longer operating time of the lower or upper group.

Therefore, in conjunction with the picking behavior described in the previous subsection, we analyze the time consumed by the robot’s picking actions. Prior to this, it is necessary to understand the kinematics of the robot. Consider a Cartesian robotic arm moving from a position \mathbf{p}_n to the next position \mathbf{p}_{n+1} ; the distance in each direction is calculated as follows:

$$\Delta \mathbf{p}_n = \mathbf{p}_{n+1} - \mathbf{p}_n \tag{1}$$

where $\mathbf{p}_n = [x, y, z]$ denotes the coordinates of the position with respect to the base frame and $\Delta \mathbf{p}_n$ represents the distance vector of the n -th position to the next. Then, the distance required to accelerate to maximum speed and decelerate to a speed of zero in each direction is calculated as follows:

$$\bar{\mathbf{s}} = \frac{\bar{\mathbf{v}} \odot \bar{\mathbf{v}}}{2\bar{\mathbf{a}}} \tag{2}$$

where $\bar{\mathbf{s}} = \langle s_i \rangle_{i=x}$ denotes the vector of acceleration/deceleration distance in three directions and $\bar{\mathbf{v}}$ and $\bar{\mathbf{a}}$ are the maximum velocity and acceleration, respectively. The required time t_i based on the distances from the current position to the next one, in each direction, is determined as follows:

$$t_i = \begin{cases} 2\sqrt{\frac{\Delta p_i}{a_{\max,i}}}, & \text{if } \Delta p_i \leq 2s_{a,i} \\ \frac{v_{\max,i}}{a_{\max,i}} + \frac{\Delta p_i - 2s_{a,i}}{v_{\max,i}}, & \text{otherwise} \end{cases} \tag{3}$$

The transitional time t_{trans} of the two positions is the maximum of the required times in each direction:

$$t_{\text{trans}} = \max(t_x, t_y, t_z) \tag{4}$$

There are four positions for the end effector of the arm during a complete harvest operation: start and end of the approach, grasp, and placement, meaning that the end effector has three translations, which are approach, extension, and retraction. In addition, the movements of the grasp and placement can also be time-consuming, which are denoted as t_{grasp} and t_{place} , respectively. Overall, the picking time for an individual fruit, denoted as t_{whole} , can be obtained using the following formula:

$$t_{\text{whole}} = \bar{t}_{\text{trans}} + \hat{t}_{\text{trans}} + t_{\text{grasp}} + \check{t}_{\text{trans}} + t_{\text{place}} \quad (5)$$

where \bar{t}_{trans} , \hat{t}_{trans} , and \check{t}_{trans} denote the translation time of approaching, extending, and retracting.

3. Multi-Arm Task Scheduling Framework

3.1. Problem Statement

Our goal is to compute an optimal task planning strategy for multi-arm robots, minimizing the idle time of each manipulator during harvesting tasks to maximize overall harvesting efficiency. To achieve this, we need to optimize the controller to dynamically adjust each manipulator's target allocation and picking sequence. This task planning problem can be viewed as a sequential decision-making process. We model it as a Markov Decision Process (MDP) and aim to solve it using reinforcement learning (RL) methods.

The main challenges of this problem stem from the mechanical coupling between the arms and their overlapping workspaces, which necessitate sequential operations of the arms (i.e., approach, extend, grasp, retract, and place). Additionally, due to the natural growth characteristics of apples, the fruit distribution within each arm's workspace is uneven. Furthermore, during the harvesting process, picking failures or accidental fruit drops may occur. Considering these factors, this paper aims to propose a dynamic task planning method suitable for multi-arm and multi-target harvesting scenarios.

3.2. Markov Game Model

3.2.1. State

The state space is defined by $\mathbf{S}_k = (\mathbf{G}, \mathbf{w}_k, \mathbf{u}_k^j)$, where $\mathbf{G} \in \mathbb{R}^{n \times 3}$ represents the coordinates of the targets with respect to the base frame; each element in $\mathbf{w}_k \in \mathbb{N}^n$ is an integer of 0 to 9, representing the state of the target at step k : 0 indicates that this target has not been visited yet, 1~4 indicates that a target has been visited and picked by Arm-1~Arm-4 respectively, 5~8 indicates that a target has been visited by an arm but failed, and 9 denotes an invalid target. $\mathbf{u}_k^j = [u_{k,1}^j, u_{k,2}^j, u_{k,3}^j]$ is a vector that contains the state variables of group j in step k , where $u_{k,1}^j$ is the previous target that group j visited using the arm represented by $u_{k,2}^j$ and $u_{k,3}^j$ is the operational time of agent j from the start to the end of step $k - 1$.

3.2.2. Action

The action for agent j at step k is defined as $a^j = [a_1^j, a_2^j]$, where a_1^j is the index of the target to harvest, ranging from 1 to n and a_2^j denotes which arm to use to visit the target a_1^j , where 0 represents the left arm and 1 represents the right arm. The action spaces of the two agents are identical.

3.2.3. State Transition

We divide the agents' picking process into several intervals, with the total operational time of the agents being the sum of these intervals. The sequential actions of the robotic

arms during picking include approach, extend, grasp, retract, and place, denoted as A, E, G, R, and P, respectively. Since the AEG stages occupy the shared degrees of freedom of the robotic arms; two arms within the same group cannot execute the AEG stages simultaneously. However, the AEG and RP stages can be executed concurrently. By partitioning the robotic arms' actions into two stages—AEG and RP—we enhance efficiency and minimize idle time.

At each state transition point, each robotic arm can choose from three action sets: AEG, RP, and idle, where idle represents the inactive state. For the two robotic arms within an agent group, there are theoretically nine possible combinations. However, due to the specific characteristics of this robot, scenarios where both arms simultaneously execute AEG or both execute RP do not occur. Therefore, there are seven valid action combinations. Each interval can thus be described by one of the following seven combinations: (AEG, RP), (RP, AEG), (idle, RP), (RP, idle), (idle, AEG), (AEG, idle), and (idle, idle), as illustrated in Figure 3.

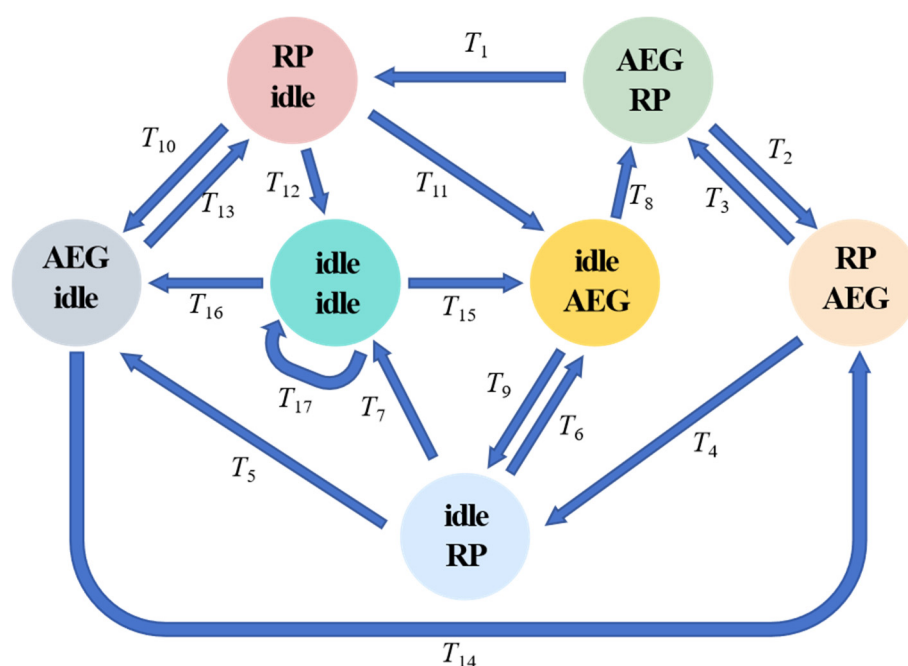


Figure 3. The cooperation of state transitions in harvesting processes.

In the figure, transitions between states are denoted by T_1 to T_{17} . For example, in transition T_1 , during the previous interval, the left arm sequentially completed the approach–extend–grasp (AEG) actions, while the right arm executed the retract–place (RP) actions. In the subsequent interval, the left arm performed the RP actions, and the right arm remained idle. In transition T_{10} , the left arm executed the RP actions in the previous interval, while the right arm was idle. In the next interval, the left arm completed the AEG actions, while the right arm continued to remain idle. Another example is transition T_{14} : the left arm completed the grasping action in the previous interval and retracted in the next interval. Meanwhile, due to interference, the right arm remained idle, while the left arm executed the AEG actions. When the left arm retracted, the longitudinal joint coupled with the left arm was released and became available for the right arm's AEG movement. This represents a typical state transition involving alternating operations between the left and right arms.

3.2.4. Reward Function

The design of the reward function leverages several key factors: the time each agent spends executing actions, the exploratory information obtained from each agent's attempts

to grasp the target fruits, and the conflict information arising between agents. The reward is structured to suppress undesirable scenarios while encouraging beneficial actions, thereby guiding the policy towards the optimization of our objectives. Specifically, since the primary goal of this study is to reduce the total operational time required for harvesting, time cost serves as a crucial component in the reward function design. By incorporating these elements, the reward function aims to balance efficiency and coordination among agents, ultimately steering the reinforcement learning policy towards minimizing the overall harvesting duration.

Additionally, let the exploration reward function be denoted as r_{exp}^i . If an agent attempts to grasp a new fruit target, a positive reward $r_{exp}^i = 0.05$ is provided, which can be obtained by observing the changes in the fruit distribution information; otherwise, $r_{exp}^i = 0$. If an agent attempts to grasp a new fruit target, a positive reward $r_{exp}^i = 0.05$ is provided, which can be obtained by observing the changes in the fruit distribution information; otherwise, $r_{exp}^i = 0$.

It should be noted that in this case, conflicts refer solely to conflicts between different agents and do not include conflicts within a single agent between its left and right arms. This is because assigning the same target to both arms within a single agent does not result in additional time costs. Since the left and right arms operate alternately, when assigned the same target, the two arms sequentially grasp the same fruit instead of waiting for each other. The conflict reward primarily considers situations where agents wait for each other. The time wasted due to the sequential grasping of the same target by the left and right arms of an agent is accounted for in the overall operational time reward.

Furthermore, let the time reward function be denoted as $r_{t_k}^i$. The working time of an agent comprises the time spent executing actions. The time reward can be expressed as:

$$r_{time}^i = \begin{cases} r_{t_{AGE}}^i & \text{if } T_k^{k+1} = T_1, T_2, T_3, T_4, T_8, T_9, T_{13}, T_{14}, \\ r_{t_{RP}}^i & \text{if } T_k^{k+1} = T_5, T_6, T_7, T_{10}, T_{11}, T_{12} \\ r_{t_{idle}}^i & \text{if } T_k^{k+1} = T_{15}, T_{16}, T_{17} \end{cases} \quad (6)$$

where $r_{t_{AGE}}^i$ represents the time required for agent III to complete the approach, extend, and grasp actions, $r_{t_{RP}}^i$ denotes the time required for agent III to complete the retract and place actions, and $r_{t_{idle}}^i$ indicates the time agent III spends in an idle waiting state. Furthermore, the overall reward function $r^i(s_k, a_k^i)$ can be defined as:

$$r^i(s_k, a_k^i) = \begin{cases} -50 & \text{if } k = k_{max} \\ 100 & \text{if all targets are harvested} \\ r_{exp}^i + r_{conf1}^i + r_{time}^i & \end{cases} \quad (7)$$

where $r^i(s_k, a_k^i)$ represents the reward received by agent I at time step k when executing action a_k^i in environment state s_k , which denotes the preset maximum number of decision steps. For instance, if k exceeds 2000, it is considered that the agent may be engaged in meaningless exploration, and the current episode is forcibly terminated with a reward of -50 .

Proximal policy optimization (PPO) is a model-free method with online on-policy and policy gradient reinforcement. It works in both discrete and continuous action spaces, which can be used in our environment. At the same time, compared with the Deep Q-Learning (DQN) method, which is also used in a discrete environment, PPO is more stable and efficient. PPO is an actor-critic type DRL algorithm; so, there are two networks in PPO: the policy network π_θ and the value network V_ϕ . π_θ is the actor network; in other words,

it maps the observations received by the agents to the actions and is parameterized by θ , which is obtained as follows:

$$\theta_{new} = \arg \max_{\theta} \mathbb{E}_{s, a \sim \pi_{\theta_{old}}} [L(s, a, \theta_{old}, \theta)] \quad (8)$$

where θ_{old} represents the old parameterized policy. L is given by

$$L(s, a, \theta_{old}, \theta) = \min(\rho_k A_k^{\theta_{old}}, \text{clip}(\rho_k, 1 - \epsilon, 1 + \epsilon) A_k^{\theta_{old}}) \quad (9)$$

where $\rho_k = \frac{\pi_{\theta}(a_k, s_k)}{\pi_{\theta_{old}}(a_k, s_k)}$ is a ratio coefficient between the updated policy and the old one and π is a small hyperparameter, which controls the policy deviation that is allowed. $A_k^{\theta_{old}}$ is the advantage function and can be calculated as follows:

$$A_k^{\theta_{old}} = \mathbb{E}_{\substack{s_{k+1} : \infty \\ a_k : \infty}} [\sum_{l=0}^{\infty} \gamma^l r_{k+l}] - V^{\theta_{old}}(s_k) \quad (10)$$

In Equation (9), the clip function is to keep ρ_k in the range $[1 - \epsilon, 1 + \epsilon]$, which can be interpreted as a regularizer of the policy network. So, Equation (9) can be reduced to the value of $\pi(a_k | s_k)$ and will be increased during the update process if $A^{\theta_{old}} \geq 0$. However, to prevent the policy from being over-updated, ρ_k will be forced to stay at $1 + \epsilon$ by the min operator. Similarly, the minimum of ρ_k will be enforced to $1 - \epsilon$ by the clip function if $A^{\theta_{old}} < 0$.

$$L(s, a, \theta_{old}, \theta) = \begin{cases} \min(\rho_k, 1 + \epsilon) A_k^{\theta_{old}}, & A_k^{\theta_{old}} \geq 0 \\ \max(\rho_k, 1 - \epsilon) A_k^{\theta_{old}}, & A_k^{\theta_{old}} < 0 \end{cases} \quad (11)$$

Another network that needs to be trained is the value function network, which is updated through the following regression:

$$\phi = \underset{\phi}{\operatorname{argmin}} \left[(V_{\phi} - R_k)^2, (\text{clip}(V_{\phi}, V_{\phi_{old}} - \epsilon, V_{\phi_{old}} + \epsilon) - R_k)^2 \right] \quad (12)$$

where $r_k = \sum_{k'-k} r_k(s_{k'}, a_{k'}, s_{k'+1})$ is the reward.

4. Experiential Results

4.1. Model Optimization and Training

We utilized Stable-Baselines3, a set of reliable implementations of reinforcement learning algorithms in PyTorch, to train and validate our algorithm [37]. In Stable-Baselines3, the policy of PPO is typically composed of a set of neural networks. These networks achieve deep feature extraction of the state space through multiple fully connected layers and map the extracted features to the action space. Therefore, the core of policy optimization lies in the training process of the neural networks.

4.1.1. Self-Attention

The feature extraction of the state space usually relies on multi-layer and fully connected networks. These networks can map input states to the action space, but their limitation lies in the difficulty of capturing the intrinsic connections between multiple related features in the input. To address this issue, self-attention has been introduced into DRL models.

The core advantage of the self-attention mechanism is its ability to enable the model to recognize and focus on the relationships between different parts of the input sequence. This mechanism allows the model to not only capture local features but also effectively integrate

global contextual information. In this way, the model can make more comprehensive and accurate judgments during the decision-making process.

In our study, self-attention meticulously computes the correlation matrix that delineates the weight relationships among distinct features. Subsequently, it employs this correlation matrix to encode the features, thereby assigning significance to each feature based on its interaction with others within the dataset.

The mathematical function of this process can be succinctly articulated as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{13}$$

where Q , K , and V represent the query, key, and value matrices, respectively. The division by $\sqrt{d_k}$ is a normalization step to stabilize the gradients during training, with $\sqrt{d_k}$ being the dimension of the key vectors. The softmax function ensures that the output weights add up to one, facilitating the interpretation of the attention weights as probabilities.

$$S_i = \frac{e^i}{\sum_j e^j} \tag{14}$$

where e^i refers to the exponential operation on the i -th element. In the function (13), Q multiplied by K results in the internal correlation of the features; then, the attention scores between the input features are obtained through the softmax function, and finally, the attention scores are multiplied by the value matrix V to obtain the final output weighted by the attention scores.

We incorporated a self-attention module into the PPO feature extraction network according to the aforementioned paradigm, as shown in Figure 4. First, the state space is flattened into a vector, which is then passed through a fully connected layer and an activation function Selu, and the output feature vector is sent to the self-attention module. Subsequently, the attention vector goes through another fully connected layer and an activation function to complete feature extraction. Finally, a fully connected function is used to map the features to the action vector space and the value vector space, respectively.

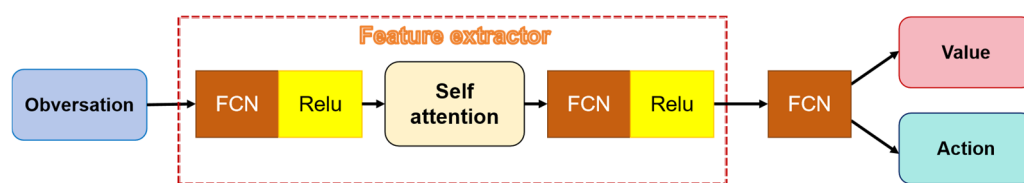


Figure 4. A deep reinforcement learning network with self-attention.

4.1.2. Training Progress

We created a Python-based harvest simulator to streamline model training. This simulator replicates the harvesting trajectories of robot groups, including target locations, arm trajectories, and collaborative actions like avoidance, waiting, and competition. This simulator is integrated into the environment to execute agent-published actions, alter the environment state, and facilitate dynamic interaction. The multi-agent task planning method using PPO is detailed in Algorithm 1, which can be used to obtain a picking sequence. Subsequently, a harvesting time is obtained by the simulator using the sequence, which is then fed back into Algorithm 1 as an input to the reward function for policy optimization. This iteration trains the task planning strategy.

In addition, a centralized training and centralized execution (CTCE) strategy is used for our policy network due to the global nature of the environment states and the shared

actions of the agents. To ensure training convergence, a gradual training approach is employed. Initially, we acquire a basic pre-trained model by training it for 1e8 steps in an environment with a static layout containing 50 targets. The training process, depicted in Figure 5, shows that the cumulative rewards and total steps per episode stabilize after training. Subsequently, we train for an additional 1e6 steps with varying target numbers. Finally, we randomly adjust the target count and environment layout at each reset, and the final policy network model is obtained after 1e8 training steps. All training steps are set according to the strategy model's ability to converge sufficiently.

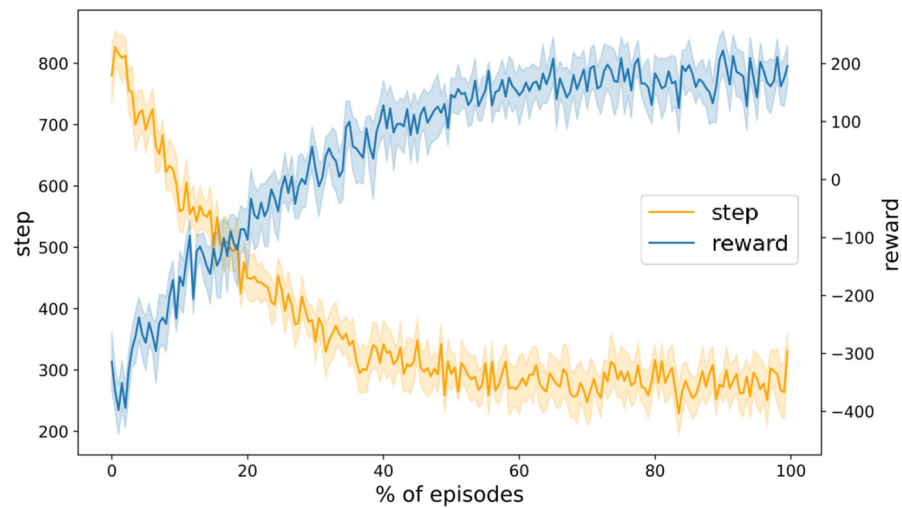


Figure 5. Steps and rewards during training.

We verified the performance of our algorithm through the following steps. First, our method was trained using a varied number of layouts generated randomly, ranging from 20 to 50, in the working space. Then, our method was evaluated by comparing it to a method that uses only four layouts. Each layout was run five times to ensure the reliability of the results. Finally, a real-world environment was built to evaluate our method and further verify its stability.

Algorithm 1 Multi-arm Task Planning and Cooperative Control

Require: the location of each target

- 1: Initialize the layout
 - 2: Initialize the state tuple \mathbf{S}_k based on the layout
 - 3: Initialize the policy π
 - 4: Initialize the number of targets n
 - 5: **while** $\Omega \leq n$ **do**
 - 6: **if** the status of Group-U is waiting, **then**
 - 7: Have \mathbf{b}_k^U decoded by the action \mathbf{a}_k^U received from π
 - 8: **if** \mathbf{b}_k^U is unavailable, **then**
 - 9: Pass
 - 10: **else**
 - 11: Execute picking \mathbf{b}_k^U using Group-U
 - 12: Set the status of Group-U as working
 - 13: **end if**
 - 14: **else**
 - 15: **if** the status of Group-D is waiting, **then**
 - 16: Have \mathbf{b}_k^D decoded by the action \mathbf{a}_k^D received from π
-

Algorithm 1 *Conts.*

```

17:         if  $\mathbf{b}_k^D$  is unavailable, then
18:             Pass
19:         else
20:             Execute picking  $\mathbf{b}_k^D$  using Group-D
21:             Set the status of Group-D as working
22:         end if
23:     end if
24: end if
25: if Group-U has finished picking, then
26:     Calculate  $t_k^U$ 
27:     Set the status of Group-U as waiting
28: end if
29: if Group-U has finished picking, then
30:     Calculate  $t_k^D$ 
31:     Set the status of Group-D as waiting
32: end if
33: Update  $r_k$  by  $t_k^U$  and  $t_k^D$ 
34: Update  $\mathbf{S}_k$ 
35: Update  $\pi$  by  $\mathbf{S}_k$  and  $r_k$ 
36: Update  $k$  and  $\Omega$ 
37: end while

```

4.2. Simulation of the PPO Framework

Our method was compared to [32], which is based on swarm intelligence heuristic search. Both methods were implemented in Python and run on a computer with an AMD Ryzen R5 2600 processor, 16 GB RAM, and Windows 10 operating system.

The parameters are listed in Table 1, where “lr” denotes the learning rate. Additionally, the parameters used in [32] were population quantity, iteration, crossover ratio, and mutation ratio, which were set to 50, 500, 0.8, and 0.3, respectively. In this context, the values of parameters ‘ γ ’ and ‘lr’ were determined through testing to achieve good convergence performance.

Table 1. The profile of parameters for the multi-arm harvesting robot.

Parameter	Value	Parameter	Value	Parameter	Value
v_x	0.25 m/s	v_y	0.1 m/s	v_z	0.30 m/s
t_{grasp}	3 s	t_{place}	2 s	-	-
γ	0.95	lr	0.0005	-	-

4.2.1. Static Environment

We initiated our evaluation with simulations to assess the performance of the random method, Tao’s proposed method [32], and our method across various static layouts. The findings are summarized in Table 2.

The random planning method markedly underperformed compared to heuristic search and reinforcement learning, suggesting the efficacy of optimization algorithms in task scheduling. Reinforcement learning outperformed heuristic search on all metrics, showcasing its superiority. Notably, reinforcement learning reduced execution time by 10.7% and 3.1% for the 25 and 50 layouts, respectively. This improvement may stem from heuristic search’s susceptibility to local minima in these scenarios, resulting in suboptimal outcomes.

The heuristic algorithm's instability for these layouts was evident in its higher variance compared to reinforcement learning across all layouts, highlighting its reduced robustness to optimization variations.

Table 2. The results of the two methods in a static environment with different layouts.

Quantity of Fruits	Method	Max.(s)	Min.(s)	Mean(s)	Var.	Calc.(s)
25	random	320.6	220.0	265.2	581.6	—
	[32]	173.60	159.3	165.3	36.7	6.3
	ours	164.5	150.8	156.7	33.5	1.0
50	random	591.1	449.1	535.1	1536.5	—
	[32]	303.6	284.9	296.0	64.5	13.1
	ours	291.2	279.8	286.2	22.7	1.16

Additionally, heuristic search proved to be more computationally intensive, and its processing time increased nearly linearly with the number of layouts. In contrast, reinforcement learning required significantly less computational time and remained unaffected by the number of layouts. This efficiency is attributed to reinforcement learning's capability to optimize policies within an autonomous model, leading to more stable planning outcomes, which is a significant advantage of this method.

4.2.2. Dynamic Environment

In the real-world harvesting environment, errors in fruit location and end-effector slippage are inevitable, often necessitating multiple attempts to successfully pick a fruit. We reflected this real-life scenario in our simulations to test the superiority of our method in a dynamic situation; to achieve this, we used the same distributions but with fruits that could require up to two attempts to be picked. Methods by Tao [32] and our method were tested three times on each layout, and total running time, average visit time, and number of missed targets were recorded. The results are detailed in Table 3.

Table 3. The results of the two methods in a dynamic environment with different layouts.

Quantity of Fruits	Round	Total Time (s)		Average Time (s)		Remaining Fruits	
		[32]	Ours	[32]	Ours	[32]	Ours
25	I	153.4	189.3	6.1	6.5	4	0
	II	181.2	185.2	7.2	6.4	4	0
	III	166.7	193.9	6.7	6.7	4	0
50	I	283.3	335.8	5.7	5.7	9	1
	II	294.4	299.5	5.9	5.1	9	2
	III	304.9	318.8	6.1	5.4	9	2

The heuristic search follows a fixed harvest sequence and does not account for picking failures, leading to fruits being overlooked. In contrast, reinforcement learning's dynamic decision-making allows it to adapt to such failures. When a pick fails, the target remains unpicked and another arm is dispatched according to the policy, significantly reducing the number of missed picks. Our results underscore the superiority of reinforcement learning over heuristic search. Notably, most fruits missed on the first attempt were successfully picked on the second try with reinforcement learning.

Additionally, due to the occurrence of repeated picking, the total picking time is no longer the sole metric for evaluating the performance of task planning algorithms. In terms of average single visit time, our method is generally superior to that of [32], although there

are instances within a scale of 25 fruits where our method lags behind that of [32]. This is partly because our method's decision-making capability is superior to that of [32], and on the other hand, the heuristic search of [32] requires recalculation for each planning, with computation time increasing linearly as the number of targets grows. Regardless of changes in target scale, our method, based on the reinforcement learning paradigm, can form a stable planning strategy through training, thus significantly reducing computational load. As the scale of picking increases, the advantage of our method over that of [32] becomes more pronounced.

Furthermore, the paper presents the joint trajectories of the four mechanical arm ends in the simulation tests, as shown in Figure 6. It can be observed that the trajectories of the ends are relatively reasonable after being scheduled by the reinforcement learning strategy, with no large spans and shorter trajectory lengths. In contrast, the trajectory quality optimized by [32] is relatively poor, exhibiting some counterintuitive detours. This is also visually manifested as relatively messy. Under 25 targets, the gap between the two algorithms may not be significant, but when considering a larger scale of fruit distribution, such as 50 targets, there is a substantial performance gap between our method and that of [32]. These findings further highlight the heuristic search's lack of robustness.

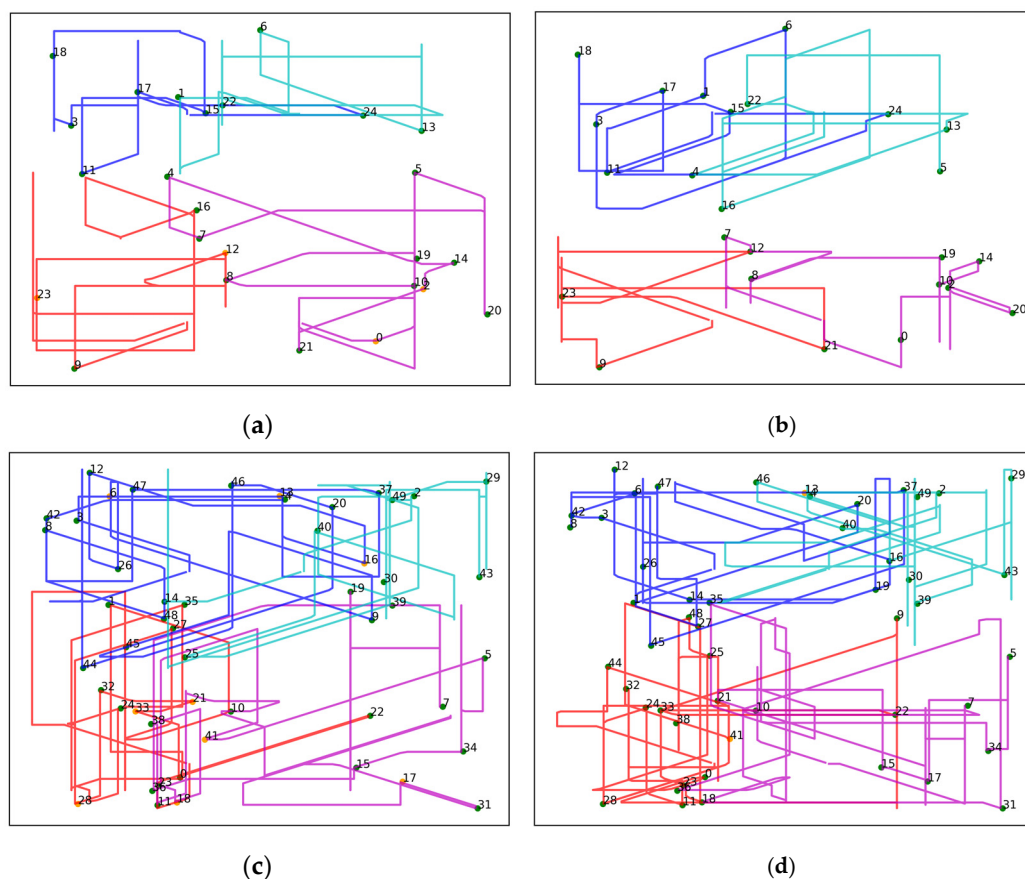


Figure 6. A deep reinforcement learning network with self-attention: (a) [32] under 25 fruits; (b) ours under 25 fruits; (c) [32] under 50 fruits; and (d) ours under 50 fruits.

Table 4 offers a detailed breakdown of the second round of 25 targets. It shows that the heuristic search-based approach had 13 instances where picking took over 10 s, mainly in the middle segment. This suggests frequent robotic arm interference, leading to significant waiting times. In contrast, our method's scheduling resulted in significantly fewer arm interference, with only six instances exceeding 10 s. Our method also has the capability to reassign arms in the event of a failed grasp, ensuring successful picking. Our method

increased attempts to pick fruits by two but still managed to reduce the total operational time by 48.2 s compared to [32]. This fully validates the superiority of our strategy over heuristic search and demonstrates its fault tolerance.

Table 4. Detailed results of the second round of the real-world environment test.

Number of Fruits	Arm of Picking		Picking Time (s)	
	[32]	Ours	[32]	Ours
1	1	2	8.2	8.4
2	3	3	8.6	12.7
3	2	1	7.4	7.2
4	4	4	8.1	8.8
5	1	2	7.6	8.0
6	2	2	7.9	8.3
7	2	1	8.7	8.1
8	3	3	14.6	16.2
9	1	2	14.7	8.5
10	4	1	12.8	8.4
11	2	4	15.3	17.1
12	1	2	16.4	16.3
13	3	3	15.2	10.8
14	1	4	13.8	8.9
15	2	1	17.1	8.7
16	4	3	10.8	16.9
17	2	2	8.6	8.7
18	1	1	8.2	7.6
19	3	2	15.3	7.8
20	2	1	14.2	6.9
21	1	2	7.6	7.2
22	3	1	18.4	7.5
23	4	4	8.3	20.7
24	3	1	8.1	9.3
25	2	2	9.2	8.1
1 (repeat)	-	1	-	7.8
3 (repeat)	-	2	-	7.4
13 (repeat)	-	4	-	8.5
24 (repeat)	-	2	-	7.9

4.2.3. Real-Word Testing

Finally, we saved the aforementioned simulation trajectories and reproduced the harvesting process on our robot to test the effectiveness and energy consumption of the algorithm when running it on a real robot. We recorded the energy consumption from the start to the end of each harvest task using a power cost; the results are as shown in Table 5:

Table 5. The results of power costs for the two methods applied to the robot.

Quantity of Fruits	Method	Power Cost (Wh)		
		Round I	Round II	Round III
25	[32]	76.7	88.9	80.3
	Ours	78.9	86.4	80.1
50	[32]	146.3	143.4	161.2
	Ours	142.6	133.1	142.0

It can be observed that although our method generally requires a longer working time than the method of [32] due to the presence of secondary grasping, it has better energy consumption. This may be because the method of [32] spends more time on mechanical

arm mutual avoidance, resulting in unnecessary energy expenditure, while our method has a longer duration of parallel operation, leading to less energy loss. This further illustrates the superiority of our method. Additionally, the simulation trajectories were perfectly reproduced on the robot, with no physical bugs occurring.

However, due to the limitations in recognition and positioning accuracy, there are still significant challenges in achieving complete perception, planning, and picking, which will be the focus of our subsequent research.

5. Conclusions

This paper explores the application of deep reinforcement learning to task scheduling for multi-arm harvesting robots. We examine the parallel, interference, and competitive behaviors exhibited by each arm during the harvesting process in a robot with coupled joints. We break down the fruit-picking process into five distinct movements and develop a formula to calculate their operational time. We then frame the task scheduling problem as a sequential decision-making process using a Markov game model. Our proposed framework is based on the deep reinforcement learning technique Proximal Policy Optimization (PPO) and incorporates a self-attention mechanism into the state feature network, enabling the model to converge rapidly.

Numerical simulations show that our method surpasses random search and heuristic approaches in terms of execution time, stability, and computational cost. Moreover, our method is capable of adapting to dynamic harvesting environments, affirming its effectiveness, which increased performance by 6.9%. Overall, our approach enhances the operational efficiency of multi-arm harvesting robots in complex scenarios. The results of fruit detection have a significant impact on the application of our task planning algorithm in real environments. Improving the accuracy of fruit recognition while mitigating the impact of inaccurate fruit positioning is a major challenge, which we will consider in the future.

Author Contributions: F.X. and Z.G. were responsible for the conceptualization, methodology, data processing, visualization, and writing—original draft and editing; T.L. was responsible for the writing—review, project administration, and funding acquisition; Q.F. and C.Z. was responsible for the project administration and funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by Science and Technology Program of Tianjin, China (23YFZCSN002 90), the Youth Research Foundation of Beijing Academy of Agriculture and Forestry Sciences, China (QNJJ202318), and the Beijing Nova Program, China (20220484023).

Data Availability Statement: The authors do not have permission to share the data.

Acknowledgments: We thank the Beijing Academy of Agriculture and Forestry Sciences for providing the necessary equipment and experimental platform for this study.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Zou, B.; Liu, F.; Zhang, Z.; Hong, T.; Wu, W.; Lai, S. Mechanization of mountain orchards: Development bottleneck and foreign experiences. *J. Agric. Mech. Res.* **2019**, *41*, 254–260.
2. Bac, C.W.; Van Henten, E.J.; Hemming, J.; Edan, Y. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *J. Field Robot.* **2014**, *31*, 888–911. [[CrossRef](#)]
3. Kang, H.; Chen, C. Fruit detection, segmentation and 3D visualisation of environments in apple orchards. *Comput. Electron. Agric.* **2020**, *171*, 105302. [[CrossRef](#)]
4. Jin, Y.; Liu, J.; Xu, Z.; Yuan, S.; Li, P.; Wang, J. Development status and trend of agricultural robot technology. *Int. J. Agric. Biol. Eng.* **2021**, *14*, 1–19. [[CrossRef](#)]

5. Kim, W.S.; Lee, D.H.; Kim, Y.J.; Kim, T.; Lee, H.J. Path detection for autonomous traveling in orchards using patch-based CNN. *Comput. Electron. Agric.* **2020**, *175*, 105620. [[CrossRef](#)]
6. Li, T.; Yu, J.; Qiu, Q.; Zhao, C. Hybrid Uncalibrated Visual Servoing Control of Harvesting Robots with RGB-D Cameras. *IEEE Trans. Ind. Electron.* **2022**, *70*, 2729–2738. [[CrossRef](#)]
7. Li, T.; Xie, F.; Zhao, Z.; Zhao, H.; Guo, X.; Feng, Q. A multi-arm robot system for efficient apple harvesting: Perception, task plan and control. *Comput. Electron. Agric.* **2023**, *211*, 107979. [[CrossRef](#)]
8. Chakraborty, S.K.; Subeesh, A.; Potdar, R.; Chandel, N.S.; Jat, D.; Dubey, K.; Shelake, P. AI-enabled farm-friendly automatic machine for washing, image-based sorting, and weight grading of citrus fruits: Design optimization, performance evaluation, and ergonomic assessment. *J. Field Robot.* **2023**, *40*, 1581–1602. [[CrossRef](#)]
9. Li, Y.; Feng, Q.; Liu, C.; Xiong, Z.; Sun, Y.; Xie, F.; Li, T.; Zhao, C. MTA-YOLACT: Multitask-aware network on fruit bunch identification for cherry tomato robotic harvesting. *Eur. J. Agron.* **2023**, *146*, 126812. [[CrossRef](#)]
10. Williams, H.; Ting, C.; Nejati, M.; Jones, M.H.; Penhall, N.; Lim, J.; Seabright, M.; Bell, J.; Ahn, H.S.; Scarfe, A.; et al. Improvements to and large-scale evaluation of a robotic kiwifruit harvester. *J. Field Robot.* **2020**, *37*, 187–201. [[CrossRef](#)]
11. Defterli, S.G. Review of robotic technology for strawberry production. *Appl. Eng. Agric.* **2016**, *32*, 301–318.
12. Tibbetts, J.H. Agricultural Disruption: New technology, consolidation, may yield production gains, job upheaval. *BioScience* **2019**, *69*, 237–243. [[CrossRef](#)]
13. Sepúlveda, D.; Fernández, R.; Navas, E.; Armada, M.; González-De-Santos, P. Robotic aubergine harvesting using dual-arm manipulation. *IEEE Access* **2020**, *8*, 121889–121904. [[CrossRef](#)]
14. Bogue, R. Fruit picking robots: Has their time come? *Ind. Robot. Int. J. Robot. Res. Appl.* **2020**, *47*, 141–145. [[CrossRef](#)]
15. Delbridge, T. Robotic strawberry harvest is promising but will need improved technology and higher wages to be economically viable. *Calif. Agric.* **2021**, *75*, 57–63. [[CrossRef](#)]
16. Xiong, Z.; Feng, Q.; Li, T.; Xie, F.; Liu, C.; Liu, L.; Guo, X.; Zhao, C. Dual-Manipulator Optimal Design for Apple Robotic Harvesting. *Agronomy* **2022**, *12*, 3128. [[CrossRef](#)]
17. Barnett, J.; Duke, M.; Au, C.K.; Lim, S.H. Work distribution of multiple Cartesian robot arms for kiwifruit harvesting. *Comput. Electron. Agric.* **2020**, *169*, 105202. [[CrossRef](#)]
18. Au, C.; Barnett, J.; Lim, S.H.; Duke, M. Workspace analysis of Cartesian robot system for kiwifruit harvesting. *Ind. Robot. Int. J. Robot. Res. Appl.* **2020**, *47*, 503–510. [[CrossRef](#)]
19. Mann, M.P.; Zion, B.; Shmulevich, I.; Rubinstein, D.; Linker, R. Combinatorial optimization and performance analysis of a multi-arm cartesian robotic fruit harvester—Extensions of graph coloring. *J. Intell. Robot. Syst.* **2016**, *82*, 399–411. [[CrossRef](#)]
20. Tang, S.Y.; Zhu, Y.F.; Li, Q.; Lei, Y.L. Survey of task allocation in multi agent systems. *Syst. Eng. Electron.* **2010**, *32*, 2155–2161.
21. Tiacci, L.; Rossi, A. A discrete event simulator to implement deep reinforcement learning for the dynamic flexible job shop scheduling problem. *Simul. Model. Pract. Theory* **2024**, *134*, 102948. [[CrossRef](#)]
22. Bouktif, S.; Cheniki, A.; Ouni, A. Traffic Signal Control Using Hybrid Action Space Deep Reinforcement Learning. *Sensors* **2021**, *21*, 2302. [[CrossRef](#)] [[PubMed](#)]
23. Lu, R.; Bai, R.; Luo, Z.; Jiang, J.; Sun, M.; Zhang, H.T. Deep Reinforcement Learning-Based Demand Response for Smart Facilities Energy Management. *IEEE Trans. Ind. Electron.* **2022**, *69*, 8554–8565. [[CrossRef](#)]
24. Horst, R.; Pardalos, P.M.; Van Thoai, N. *Introduction to Global Optimization*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2000.
25. Floudas, C.A.; Lin, X. Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Ann. Oper. Res.* **2005**, *139*, 131–162. [[CrossRef](#)]
26. Li, W.; Ding, Y.; Yang, Y.; Sherratt, R.S.; Park, J.H.; Wang, J. Parameterized algorithms of fundamental NP-hard problems: A survey. *Hum.-Centric Comput. Inf. Sci.* **2020**, *10*, 1–24. [[CrossRef](#)]
27. Sasaki, T.; Biro, D. Cumulative culture can emerge from collective intelligence in animal groups. *Nat. Commun.* **2017**, *8*, 15049. [[CrossRef](#)] [[PubMed](#)]
28. Li, Z.; Barenji, A.V.; Jiang, J.; Zhong, R.Y.; Xu, G. A mechanism for scheduling multi robot intelligent warehouse system face with dynamic demand. *J. Intell. Manuf.* **2020**, *31*, 469–480. [[CrossRef](#)]
29. Zhang, K.; He, F.; Zhang, Z.; Lin, X.; Li, M. Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach. *Transp. Res. Part C Emerg. Technol.* **2020**, *121*, 102861. [[CrossRef](#)]
30. Panerati, J.; Gianoli, L.; Pincirolino, C.; Shabah, A.; Nicolescu, G.; Beltrame, G. From swarms to stars: Task coverage in robot swarms with connectivity constraints. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 7674–7681.
31. Chen, J.; Zhang, Y.; Wu, L.; You, T.; Ning, X. An adaptive clustering-based algorithm for automatic path planning of heterogeneous UAVs. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 16842–16853. [[CrossRef](#)]
32. Li, T.; Qiu, Q.; Zhao, C. Task planning of multi-arm harvesting robots for high-density dwarf orchards. *Nongye Gongcheng Xuebao/Trans. Chin. Soc. Agric. Eng.* **2021**, *37*, 1–10.

33. Tian, Y.T.; Yang, M.; Qi, X.Y.; Yang, Y.M. Multi-robot task allocation for fire-disaster response based on reinforcement learning. In Proceedings of the 2009 International Conference on Machine Learning and Cybernetics, Baoding, China, 12–15 July 2009; IEEE: Piscataway, NJ, USA, 2009; Volume 4, pp. 2312–2317.
34. Wilson, A.; Fern, A.; Ray, S.; Tadepalli, P. Multi-task reinforcement learning: A hierarchical bayesian approach. In Proceedings of the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; pp. 1015–1022.
35. Sun, L.; Yu, X.; Guo, J.; Yan, Y.; Yu, X. Deep reinforcement learning for task assignment in spatial crowdsourcing and sensing. *IEEE Sens. J.* **2021**, *21*, 25323–25330. [[CrossRef](#)]
36. Choudhury, S.; Gupta, J.K.; Kochenderfer, M.J.; Sadigh, D.; Bohg, J. Dynamic multi-robot task allocation under uncertainty and temporal constraints. *Auton. Robot.* **2021**, *46*, 231–247. [[CrossRef](#)]
37. Antonin, R.; Ashley, H.; Adam, G.; Anssi, K.; Maximilian, E.; Noah, D. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.* **2021**, *22*, 268:1–268:8.

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