Let Robots Watch Grass Grow: Optimal Task Assignment for Automatic Plant Factory

Zhengzhe Xiang, Xizi Xue, Yuanyi Chen, *Member, IEEE*, Schahram Dustdar, *Fellow, IEEE*, and Minyi Guo, *Fellow, IEEE*

***Abstract*—Modularized plant factories, characterized by machines executing intelligent control requests to automatically take care of crops, have emerged as a sustainable agricultural paradigm, garnering the attention of Internet-of-Things and agricultural researchers for their production stability and energy efficiency. However, the diversity and pluralism of the plant factory components make it difficult to cooperate and produce crops with better qualities. Therefore, appropriate resource allocation and task scheduling strategies become the key points to optimize the quality of production in the factories by immediately telling which component is more suitable to do what in taking care of the crops. To address this challenge, this paper investigates how the machines of the factory can use their unique services and resource to help improve the crops' quality and model the machine cooperation as an online decision-making problem. An α-competitive approach called ONATS is designed based on the transformation of the original problem, and the experiments show that the proposed algorithm is superior to the baselines. Additionally, this paper explores the impact of different system configurations on the proposed method and shows that the proposed approach has broad applicability.**

***Index Terms*—Service deployment, plant factory, resource allocation, task assignment.**

I. Introduction

MAYBE you are now reading this manuscript after a simple but nutritious meal, but we have to admit that our world is still starving. According to *The State of Food Security and Nutrition in the World*[1] reported by the United Nations, the food insecurity problem has been deteriorating continuously since the COVID-19 pandemic, Russia-Ukraine war and their consequences keep on challenging the recovery of the world's economy. What's more, the war between two of the biggest food vendors in the world brings in more uncertainty to the food supply chains. As a consequence, 3.1 billion people have to face the shortage of the cheapest healthy diet every day. On the other hand, the population of our world is predicted to exceed 9×10^{10} in the coming 30 years, and it becomes more difficult to feed the world if we fail to produce more with the shrinking environmental resources. Therefore, "how to grow more with less", namely *sustainable developing*, emerges as a vital issue that the scientists have to face [1]. Among the proposed solutions to this problem, the *plant factory* is regarded as a promising option. Plant factories are fully-closed systems with an artificial environment to grow crops all year round, controlled by automated environmental facilities [2]. In contrast to traditional agriculture, plant factories rely on internal devices rather than external factors to determine the environmental conditions. This unique approach allows for vertical growth of crops, maximizing the utilization of space and resulting in higher crop yields per unit of land [3]. Additionally, the implementation of closed-loop irrigation systems enables the recirculation and reuse of water within the system, thereby reducing water consumption [4] and dependence on pesticides and herbicides. For instance, the researchers and engineers of SANANBIO[2] have successfully realized the running of the first $5 \times 10^3 \mathrm{m}^2$ artificial anoectochilus roxburghii plant factory in China. The annual harvest of this factory is comparable to that of the entire Nanjing County, which contributes the most product in Fujian Province.

While the pursuit of massive and automatic production remains a goal, the high construction costs and inflexibility of *integrated plant factories* have hindered their adoption in recent years, especially given the immaturity of many involved techniques. In contrast, the *modularized plant factory* has attracted attention from practitioners with its concept of "replaceable components". In this concept, each component is responsible for its own functions, and specific "adapters" (often in the form of different types of robots) will work as lubricants to coordinate these components. By monitoring the running status of the components through APIs provided by their built-in Internet-of-Things (IoT) devices, the plant factory management system can send requests to the components and adapters to take collaborative actions to serve the plants. In this way, the modularized plant factory can provide customized growing capacity at a lower cost with acceptable crop output.

Received 3 October 2023; revised 26 July 2024; accepted 10 September 2024. Date of publication 17 September 2024; date of current version 5 June 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62102350 and Grant 62072402, in part by the Natural Science Foundation of Zhejiang Province under Grant LQ21F020007, and in part by the Top Young Scholar of Zhejiang Province "Ten Thousands Talent Program". Recommended for acceptance by Y. Jiang. *(Corresponding author: Yuanyi Chen.)*

Zhengzhe Xiang and Yuanyi Chen are with Hangzhou City University, Hangzhou 310015, China (e-mail: xiangzz@hzcu.edu.cn; chenyuanyi@hzcu.edu.cn).

Xizi Xue is with Zhejiang University, Hangzhou 310027, China (e-mail: xuexizi@zju.edu.cn).

Schahram Dustdar is with Distributed Systems Group, TU Wien, 1040 Vienna, Austria, and also with Universitat Pompeu Fabra (UPF), 08002 Barcelona, Spain (e-mail: dustdar@dsg.tuwien.ac.at).

Minyi Guo is with Emerging Parallel Computing Center, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: guo-my@cs.sjtu.edu.cn).

Digital Object Identifier 10.1109/TSUSC.2024.3462447

[1] https://www.fao.org/publications/sofi/2022/en/

[2] https://www.sananbio.com/

It is apparent that modularized plant factories require high processing efficiency. However, the problem of assigning requests to multiple adapters in the factory is a challenging task, particularly when the adapters are not homogeneous due to their different service initiation times. For instance, some robots are capable of transplanting seedlings while others can only perform picking tasks using their mechanical arms. Clearly, *the shortcomings in the management of modularized plant factories will impede their operations, compromise the quality of crops, and ultimately hinder progress towards sustainable development*. Hence, it is essential to address the multi-robot task assignment problem, especially when robot actions are resource-intensive and their resources are limited. However, a comprehensive literature review on resource allocation and task scheduling optimization in MEC-based service systems revealed that a majority of these studies did not focus on plant factories. Moreover, there is a noticeable lack of precise models that can describe the specific entities and their effects in modular plant factory scenarios. Consequently, in this paper, we propose a tailored mathematical model that accounts for the unique characteristics of modularized plant factories. We also conduct a series of experimental analyses and discussions to demonstrate the effectiveness of our proposed model. The contributions of this paper are listed as follows:

1) We investigated the use of intelligent adapters in modularized plant factories and proposed a model to represent their behaviors and corresponding qualities based on the contribution they provide to the system.

2) We addressed the online request assignment problem in modularized plant factories and developed an approach based on multi-knapsack research to ensure that the crops receive α-competitive care.

3) To evaluate the effectiveness of our proposed methodology, we conducted comprehensive experiments and compared the results to existing baselines. Furthermore, we examined the impact of different system configurations on the experimental outcomes.

In addition, due to the fact that related problems such as sluggish response and execution failures may also occur when the computation complexity exceeds the reach of the adapter's own equipment or when the instruction transmission from the cloud costs too much time. Therefore, the multi-access edge computing (MEC) technique, which optimizes the use of mobile resources and wireless networks to provide good service quality with edge servers deployed close to the fields [5], [6], [7], [8], is utilized here as well to empower the plant factory by setting the computing unit in the factory.

The paper is structured as follows: Section II provides a simple example to illustrate the target problem. Section III presents an overview of related research. Section IV describes the concepts and explains how the system works. Section V introduces our approaches. Section VI presents the experimental results, including the factors that affect our algorithms and the comparison with baselines. Section VII concludes the contribution and outlines the future work.

Fig. 1. Robots work as adapters in a plant factory.

II. Motivation and Scenario

In this section, we will discuss the motivation behind the research and present a scenario that exemplifies the target problem. To illustrate the issue clearly, we will use a simple plant factory with multiple robots working within it as an example. This approach will enable us to identify possible optimization challenges before delving into a detailed problem description.

Actually, in the robot-aided modularized plant factory illustrated in Fig. 1, the main job of the robots is to take care of the crops like human beings — sometimes they need to turn over the leaves to ensure the preset cameras can observe the crops all-around, sometimes they need to heal the crops with the correct pesticide in their containers, some times they need to prune the branches to reduce the effects of water transpiration, and sometimes they need to harvest the crops and start a new round of seeding. Though required materials or equipment are different in these situations, but the basic logic is the same — consume what a robot has to do what a robot can. This is easy to understand because the volume of the containers is limited, and the manipulators of a robot will need maintenance after a fixed period of time.

Taking a plant factory with 3 robots as an example. For every robot, it will be responsible for supplementing different growing resources. For example, robot #1 can provide three services: loosening soil, spraying pesticides, and pruning vines when it receives the instruction for related resources; robot #2 can provide two services: loosening soil and harvesting crops when corresponding requirements are sensed by the management system and the tasks are assigned to it; robot #3 is more flexible because it can provide all the four services of loosening soil, spraying pesticides, pruning vines and harvesting crops. Therefore, in this case, we can represent the resources with

R_1={C_1: 5, C_2: 3, C_3: 4, C_4: 0},R_2 = {C_1: 4, C_2: 0, C_3: 0, C_4: 5},R_3 = {C_1: 6, C_2: 4, C_3: 3, C_4: 5} to show the current resources they have, where C_1, C_2, C_3, C_4 represent their capacities (or maximum resource) in soil loosening, pesticide spraying, vine pruning, and harvesting services respectively, the numerical value indicates how many units of care resources can be provided. What's more, as the robots are heterogeneous, their service qualities will also be different — different robots will provide different benefits even if they use the same kind of service. Here we simply assume that the value function of each service of each robot is $g(x)_{i,n} = A_{i,k}B_n x$, where A represents the quality coefficient of service k provided by the robot #i, B_n represents the quality coefficient influenced by the n-th request and x represents the amount of resource. The service quality factor vectors for each robot are $A_1 = [1, 2, 3, 4]$, $A_2 = [2, 2, 1, 3]$, $A_3 = [3, 2, 3, 1]$. In addition, because each request is different in urgency and value, it also leads to different benefits. For example, assume there were 3 requests arriving in chronological order. The first request requires 3 units of ripping service C_1, and along with request 1 comes the coefficient $B_1 = 2$, so request #1 can be represented as Q_1 = {C_1: 3, G_B: 2}. In the same way, we can use Q_2 = {C_4: 2, G_B: 1},Q_3 = {C_1: 5, G_B: 5} to represent the two requests that come later. At this point, we have multiple allocation methods. Since the service quality and service efficiency provided by different robots are different, the benefits obtained by different schemes are also not the same. In scheme 1, we can assign Q_1 to robot R_1, Q_2 to robot R_2, and Q_3 to robot R_3, so the benefit that can be obtained is 87; In scheme 2, we can assign Q_1 to robot R_3, Q_2 to robot R_2, and Q_3 to robot R_1, so the yield value 49 can be obtained. It is clear that $87 > 49$, and scheme 1 is better than scheme 2. In addition, we can split requests and assign them to multiple bots serving together, which leads to more different schemes.

In this paper, our goal is to maximize the total return of the plant factory by optimizing the allocation and scheduling strategies to make appropriate decisions.

III. Related Work

Resource allocation and task scheduling strategies are important issues that widely discussed in service system research to support the needs of different businesses in today's IoT industry. They will dramatically affect the efficiency of resource use and user service experience, especially when resources and requirements are also multi-dimensional.

In general, researchers would like to translate the target strategy generation problem into some optimization problems and then solve them to obtain accurate or approximate solutions. For instance, Cao et al. [9] used hierarchical clustering to modify the Two_Arch2 algorithm and solved the resource allocation problem of the fog-cloud system in the Internet of Vehicles environment. Yu et al. [10] converted the non-convex objective function into a suitable convex approximation based on the SCA algorithm, then iteratively solved the optimization problem to optimize the service provisioning in the UAV MEC system. Priya et al. [11] proposed a fuzzy-based Multidimensional Resource Scheduling (MRS) algorithm, which effectively improved resource scheduling efficiency and average success rate via using Multidimensional Queuing Load Optimization algorithm. Yang et al. [12] investigated scenarios such as task scheduling and virtual machine resource allocation, modeled them as online multidimensional knapsack problems, and developed two algorithms called LinRP and ExpRP for online admission decisions. Halabian [13] presented a fully distributed solution for the resource allocation problem by forming a resource auction between the slices and the data centers based on the notion of auction theory. Rahman et al. [14] raised a non-cooperative game theory-based approach to assign each request for a sensing task to a single node in the cluster. Khalil et al. [15] proposed a nature-inspired meta-heuristic algorithm to guarantee reliable task distribution and improve the energy efficiency and stability period of the network.

Currently, some artificial intelligent models and learning-based methods are also adopted to empower the strategy generation problem. For example, Gai et al. [16] combined QoE with RL to create pre-stored cost mapping tables for optimal resource allocation. Ding et al. [17] proposed a Q-learning-based dynamic task scheduling framework QEEC, applying incentives to reward the assignments that can minimize task response time and maximize each server's CPU utilization. Huang et al. [18] introduced a deep adversarial imitation reinforcement learning framework called AIRL for scheduling time-sensitive cloud jobs.

These studies clarify the basic outlines of resource allocation and task scheduling optimization in MEC-based service systems. However, they didn't focus on the scenario of plant factories so no clear models are constructed for the entities and their corresponding abilities in it.

IV. System Model and Problem Description

In this paper, we mainly consider the task assignment problem of the various robots and their collaborations in the modularized plant factory, and we will introduce the related entities and their interactions in this section. Based on this, we will go further to reveal the relationship between the task assignment strategy and the final harvest with an approximate quantitative model and try to optimize the final harvest by online task assigning. The used notations in this paper are shown in Table I for simplification.

A. System Model

Suppose there are M robots working as intelligent adapters in the modularized plant factory, which can be denoted as $\mathcal{R} = \{r_1, r_2, r_3, \ldots, r_M\}$. For every robot r_m, it can provide as much as K types of services of the service set $\mathcal{S} = \{s_1, s_2, s_3, \ldots, s_K\}$. Different types of services have different functionalities. For example, the service s_1 may be "watering" while the service s_2 may be "trimming". As providing specific functions to accomplish certain tasks will be resource-consuming, here we use a vector $\boldsymbol{c}_m = \{c_{m,1}, c_{m,2}, \ldots, c_{m,K}\}$ to represent the total available resource the robot r_m has for different types of services. For example, when the service s_1 is "watering" and s_2 is "trimming" as mentioned above, here $c_{m,1} = 500$ and $c_{m,2} = 10$ may

TABLE I
SYMBOL DESCRIPTION

Symbols	The physical meaning of the notations
M	number of adapters (or robots)
K	number of service types
J	the number of virtual "knapsacks", equals to $M \times K$
$\mathcal{R}$	set of adapters
$\mathcal{S}$	set of services
$\mathcal{N}$	a counter set of the requests
$c_{m,k}$	total available resource that robot r_m has for service k
$\boldsymbol{q}^{(n)}$	the n-th control request
$s^{(n)}$	indicate which service is required by request $\boldsymbol{q}^{(n)}$
$\delta^{(n)}$	recommended max resource for $\boldsymbol{q}^{(n)}$
$\eta_{m,k}^{(n)}$	assignment constraint for adapter $r_{m,k}$ in $\boldsymbol{q}^{(n)}$
$\gamma_{m,k}^{(n)}(\cdot)$	utility or profit function for adapter $r_{m,k}$ in request $\boldsymbol{q}^{(n)}$
$y_{m,k}^{(n)}$	resource amount that the adapter r_m used for service s_k
$\boldsymbol{\mathcal{R}}^{(n)}$	set of adapters which can help to run $s^{(n)}$
$\mathscr{A}^{(n)}$	request-aware parameter in the utility function
$\mathscr{B}_{m,k}$	capabilities of $r_{m,k}$ in providing different services
$\boldsymbol{y}^{(n)}$	decision variable of $\boldsymbol{q}^{(n)}$
$\boldsymbol{x}^{(n)}$	vectorized decision variable transformed from $\boldsymbol{y}^{(n)}$
$\phi(w)$	function that quantifies the marginal cost of putting items whose total weight is w
$w_j^{(n)}$	total weight of the virtual knapsack j
$\mathcal{L}$	infimum of the partial derivative of function $\gamma_{m,k}^{(n)}(\cdot)$
$\mathcal{U}$	supremum of the partial derivative of function $\gamma_{m,k}^{(n)}(\cdot)$
α	the competitive ratio of ONATS
ρ	ratio of $\mathcal{U}$ and $\mathcal{L}$
β_j	segmentation point of the function $\phi(w)$

describe the fact that r_m can hold at most 500mL water and provide 10 hours of trimming. Obviously, as the robots may be heterogeneous and have various structures, it is not necessary to force them to provide equal services. So we use $c_{m,k} = 0$ to mark the disability of r_m if it is not suitable to accomplish the task declared by service s_k.

As the modularized plant factory will monitor and analyze the environment all the time and try to control it as well as the growing of the crops, the decisions made by the plant factory system will be generated continuously. The decisions will act as constantly coming requests that tell what and how to do. Here we characterize a coming control request by a tuple $\boldsymbol{q}^{(n)} = (s^{(n)}, \delta^{(n)}, \{\eta_{m,k}^{(n)}\}_{m=1,k=1}^{M,K}, \{\gamma_{m,k}^{(n)}(\cdot)\}_{m=1,k=1}^{M,K})$. The index n shows the order of request $\boldsymbol{q}^{(n)}$, and we denote $\mathcal{N}$ as the counter of the requests where $n = |\mathcal{N}|$ will always be the index of the last request this time. Based on the above points, if we use $y_{m,k}^{(i)}$ to denote the resource amount that the adapter r_m will use for service s_k in processing request $\boldsymbol{q}^{(i)}$, we will have the following constraints:

$$\sum_{i=1}^{n} y_{m,k}^{(i)} \leq c_{m,k} \tag{1}$$

$s^{(n)} \in \mathcal{S}$ indicates which service is required by request $\boldsymbol{q}^{(n)}$. According to the value of $\{c_{m,k}\}_{m=1,k=1}^{M,K}$, we will easily pick out the possible adapters that can help to run $s^{(n)}$ by constructing the set $\boldsymbol{\mathcal{R}}^{(n)} = \{r_m | r_m \in \mathcal{R}, c_{m,\xi(n)} > 0\}$ for every coming request $\boldsymbol{q}^{(n)}$. Here, $\xi(n)$ is the `lookup` function that indicates which type the request $\boldsymbol{q}^{(n)}$ requires:

$$\xi(n) = k, \text{if } s^{(n)} = s_k \tag{2}$$

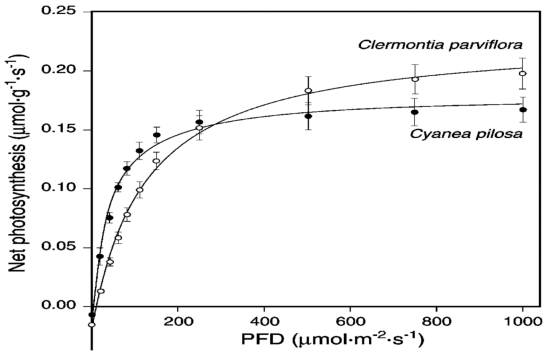

Fig. 2. Static photosynthetic light responses of Cyanea pilosa and Clermontia parviflora [19].

The tuple $(\delta^{(n)}, \{\eta_{m,k}^{(n)}\}_{m=1,k=1}^{M,K}, \{\gamma_{m,k}^{(n)}(\cdot)\}_{m=1,k=1}^{M,K})$ describes how the used resources of the robots can contribute to the final utility. First, the $\delta^{(n)}$ reveals the recommended maximum resource that can be occupied for $\boldsymbol{q}^{(n)}$ considering the universal marginal effect, and $\delta^{(n)}$ will constrain the value of $y_{m,k}^{(n)}$ with:

$$\sum_{m=1}^{M}\sum_{k=1}^{K} y_{m,k}^{(n)} \leq \delta^{(n)} \tag{3}$$

Second, the $\eta_{m,k}^{(n)}$ addresses the preference of choosing suitable adapters by constraining the allocation strategy with

$$0 \leq y_{m,k}^{(n)} \leq \eta_{m,k}^{(n)} \tag{4}$$

In the most general case where all the adapters are treated equally, the preference will be set by $\eta_{m,k}^{(n)} = c_{m,k}$ in the request. And in the case where some adapters are deemed not qualified, the preference will be described by setting $\eta_{m,k}^{(n)} =$ 0in the request tuple. $\gamma_{m,k}^{(n)}(\cdot)$ is the utility or profit function that describes the relationship between the resource allocated to r_m in accomplishing the task declared by s_k and the utility (or profit) gain it will contribute to the final harvest. Usually, the utility function $\gamma_{m,k}^{(n)}(\cdot)$ is given by empirical study, which shows a non-decreasing but marginal diminishing relationship.

The work of [19] shows the observations on clermontia parviflora and cyanea pilosa in Fig. 2: with the increasing of photon flux density, the net photosynthesis rate (which reflects the growing status of the crops) increases as well, but the increasing changes from fast to slow. Based on this, here we use a family of parameterized functions in the pattern of $\gamma_{m,k}^{(n)}(x) = \mathscr{A}^{(n)} \ln(1 + \mathscr{B}_{m,k} \cdot x)$ to quantify the relation approximately. The approximation is derived from many existing works [19]. In this series of functions, the parameters $\{\mathscr{A}^{(n)}\}$ that bounded by $\mathscr{A}_{\min} \leq \mathscr{A}^{(n)} \leq \mathscr{A}_{\max}$ are request-aware to show the difference between the control requests so that will have different effectiveness in working. On the other hand, the parameters $\{\mathscr{B}_{m,k}\}$ that bounded by $\mathscr{B}_{\min} \leq \mathscr{B}_{m,k} \leq \mathscr{B}_{\max}$ address the capabilities of the adapters in providing different services. It is worth noting that this pattern is not fixed, the followers can easily replace

it with other patterns if they want to extend the model in their applications.

B. Problem Formulation

With the system described above, the target problem of running such a modularized plant factory will be quite clear: for the continuously produced requests, how to find an appropriate strategy that determines how much resource will be allocated to maximize the total utility online. Namely, if there exist n-1 requests and now comes the n-th control request $\boldsymbol{q}^{(n)} = (s^{(n)}, \delta^{(n)}, \{\eta_{m,k}^{(n)}\}_{m=1,k=1}^{M,K}, \{\gamma_{m,k}^{(n)}(\cdot)\}_{m=1,k=1}^{M,K})$, how to set the values of $y_{m,k}^{(n)}$ when the objective and constraints are represented with:

$$P_1: \quad \max_{\boldsymbol{y}^{(n)}} \quad \sum_{i=1}^{n}\left(\sum_{m=1}^{M}\sum_{k=1}^{K}\gamma_{m,k}^{(i)}(y_{m,k}^{(i)})\right) \tag{5}$$

$$\text{s.t.} \sum_{i=1}^{n} y_{m,k}^{(i)} \le c_{m,k}, \forall m,k \tag{6}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{K} y_{m,k}^{(n)} \le \delta^{(n)} \tag{7}$$

$$0 \le y_{m,k}^{(n)} \le \eta_{m,k}^{(n)}, \forall m,k \tag{8}$$

It is worth noting that the formulation above doesn't mean that the problem is offline because the system will not stop running after generating the values of $\boldsymbol{y}^{(n)}$ until all the adapters are out of resources.

V. Online Task Assignment for Robots

A. Problem Transformation

To solve the above problem, we may first try to use some transforms to reduce it. Obviously, the decision variable $\boldsymbol{y}^{(n)}$ is described by an allocation matrix:

$$\boldsymbol{y}^{(n)} = \begin{bmatrix} y_{1,1}^{(n)} & y_{1,2}^{(n)} & \cdots & y_{1,k}^{(n)} & \cdots & y_{1,K}^{(n)} \\ y_{2,1}^{(n)} & y_{2,2}^{(n)} & \cdots & y_{2,k}^{(n)} & \cdots & y_{2,K}^{(n)} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{M,1}^{(n)} & y_{M,2}^{(n)} & \cdots & y_{M,k}^{(n)} & \cdots & y_{M,K}^{(n)} \end{bmatrix} \tag{9}$$

but we can find this matrix is equivalent to the vector $[y_{h^{-1}(1)}^{(n)}, y_{h^{-1}(2)}^{(n)}, \ldots, y_{h^{-1}(J)}^{(n)}]$ if we construct a mapping h: $\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ where $h(m,k) = (m-1)\cdot K + k$ and its inverse mapping h^{-1}: $\mathbb{N}\to\mathbb{N}\times\mathbb{N}$ where $h^{-1}(j) = (\lfloor j/K \rfloor + 1, j-1 \bmod K + 1)$. For simplification, here we denote the vectorized decision variable with $\boldsymbol{x}^{(n)}$ and let $\boldsymbol{x}^{(n)} = [x_1^{(n)}, x_2^{(n)}, \ldots, x_J^{(n)}]$ where $x_j^{(n)}$ stands for $y_{h^{-1}(j)}^{(n)}$ in $\boldsymbol{y}^{(n)}$ and the value of J can be obtained by $J = M \times K$. In this way, the target problem is reduced to:

$$P_2: \quad \max_{\boldsymbol{y}^{(n)}} \quad \sum_{i=1}^{n}\left(\sum_{j=1}^{J}\gamma_j^{(i)}(x_j^{(i)})\right) \tag{10}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_j^{(i)} \le c_j, \forall j \in \{1,2,\ldots,J\} \tag{11}$$

$$\sum_{j=1}^{J} x_j^{(n)} \le \delta^{(n)} \tag{12}$$

$$0 \le x_j^{(n)} \le \eta_j^{(n)}, \forall j \in \{1,2,\ldots,J\} \tag{13}$$

Obviously, the problem is now reduced to a one-dimensional form. Then, we would like to show that the reduced problem will be a special case of the online bin-packing or knapsack problem: In fact, if we treat the J adapter-service pairs as J "knapsacks" where c_j is the capacity of the j-th "knapsack" and $\delta^{(n)}$ is the size of the coming n-th "item", then the request $\boldsymbol{q}^{(n)}$ can also be treated as a coming "item" in the knapsack problem. When the n-th "item" comes, it will be divided into at most J parts so that every part can be put into one of the J "knapsacks". Obviously, as the "knapsakcs" are not infinite, the j-th "knapsack" will only provide $\eta_j^{(n)}$ of its capacity to the n-th "item". Finally, when $x_j^{(n)} (\le \eta_j^{(n)})$ is provided, $\gamma_j^{(i)}(x_j^{(i)})$ will describe the "value" of this decision.

B. Analysis and Approach

Usually, a sort of threshold function-based strategy is adopted in such online decision-making problems [20], [21], [22]. As shown in these works, the strategy threshold function-based uses a function $\phi(w)$ to quantify the marginal cost of putting items whose total weight is w so that the solution is generated by maximizing the difference of $\phi(\cdot)$ before and after packing the coming item. Similarly, here we also take the idea of generating solutions by observing the difference of related threshold function, namely the value of function $\Omega(\boldsymbol{x}^{(n)}) = \sum_{j=1}^{J}\int_{w_j^{(n)}}^{w_j^{(n)}+x_j^{(n)}}\phi_j(u)du$ to determine how much resource will be allocated if $w_j^{(n)}$ is the used size before the n-th request. In this work, the threshold function is designed [23] as follows for each j in $\boldsymbol{y}^{(n)}$:

$$\phi_j(w) = \begin{cases} \mathcal{L} & , w \in [0, \beta_j] \\ \frac{\mathcal{U}-\mathcal{L}}{\exp(\alpha)-\exp(\frac{\alpha}{\alpha-1})}\exp(\frac{\alpha}{c_j}w) + \frac{\mathcal{L}}{\alpha} & , w \in [\beta_j, c_j] \end{cases}$$

where the $\mathcal{L}$ and $\mathcal{U}$ are the infimum and supremum of the value function:

$$\mathcal{L} = \min_{\substack{1\le j\le J,\\ 1\le i\le n}} \frac{\partial\gamma_j^{(i)}(y_j^{(i)})}{\partial y_j} = \frac{\mathscr{A}_{\min}\mathscr{B}_{\min}}{\mathscr{B}_{\min}\delta_{\max}+1}$$

$$\mathcal{U} = \max_{\substack{1\le j\le J,\\ 1\le i\le n}} \frac{\partial\gamma_j^{(i)}(y_j^{(i)})}{\partial y_j} = \mathscr{A}_{\max}\mathscr{B}_{\max} \tag{14}$$

and α is the constant whose value is described by the solution of the following equation

$$\alpha - 1 - \frac{1}{\alpha - 1} = \ln\frac{\alpha\cdot\rho - 1}{\alpha - 1} \tag{15}$$

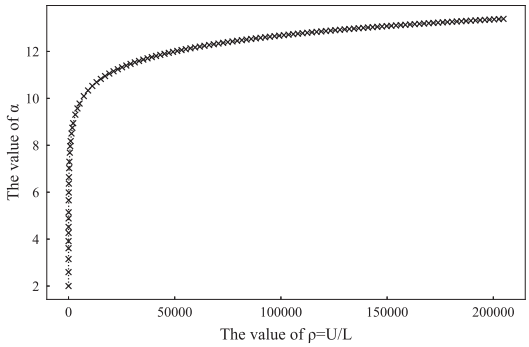

Fig. 3. The relationship between ρ and α.

Fig. 4. Raspberry Pi robot car in a modularized plant factory.

if we denote $\rho = \mathcal{U}/\mathcal{L}$, and $\beta_j = c_j/(\alpha - 1)$. The relationship between ρ and α is shown in Fig. 3, it can be seen that the competitive ration will not be too large in practice.

With this piecewise threshold function, it can be proved that compared with the global offline optimum, an α-competitive (which means the derived objective is at least $1/\alpha$ times than the global offline optimum) will be obtained by optimizing $\Psi(\boldsymbol{x}^{(n)})$ where

$$\Psi(\boldsymbol{x}^{(n)}) \triangleq \sum_{i=1}^{n} \left(\sum_{j=1}^{J} \gamma_j^{(i)}(x_j^{(i)}) \right) - \Omega(\boldsymbol{x}^{(n)}) \tag{16}$$

immediately when every request $\boldsymbol{q}^{(n)}$ arrives. It is worth noting that the function $\Omega(\boldsymbol{x}^{(n)})$ can be represented with equation (17) shown at the bottom of the this page:

In summary, after transforming and reducing the target problem and using the idea of threshold function, the online decision-making strategy can now be described with the Algorithm 1, whose name is ONATS.

Algorithm 1: ONATS.

Input: The threshold functions $\{\phi(\cdot)\}$ and the utility functions $\{\gamma_j(\cdot)\}$ for adapter-service pairs where $j \xleftarrow{h^{-1}} (m, k)$; the lower and upper bounds $\mathcal{L}$ and $\mathcal{U}$ of $\{\gamma_j(\cdot)\}$

Output: the resource allocation strategy for the coming request $\boldsymbol{y}^{(n)}$

initialize utilization $w_j^{(1)} \leftarrow 0, 1 \le j \le J$
while *control request* $\boldsymbol{q}^{(n)}$ *comes into the system* **do**
  $\boldsymbol{v}^{(n)} = \underset{\boldsymbol{x}^{(n)}}{\operatorname{argmax}} \Psi(\boldsymbol{x}^{(n)})$
end while
for $j = 1$ *to* J **do**
  update $w_j^{(n+1)} \leftarrow w_j^{(n)} + v_j^{(n)}$
  $(m, k) \leftarrow h^{-1}(j)$
  $y_{m,k}^{(n)} \leftarrow v_j^{(n)}$
end for
return *the allocation decision* $\boldsymbol{y}^{(n)}$

VI. EXPERIMENTS AND ANALYSIS

A. Preliminary

In this section, we've constructed a mini modularized plant factory to evaluate the proposed method. The system consists of 10 Raspberry Pi[3] robot cars, simulating the application of adapters in real-world planting operations. Each robot car is equipped with multiple spraying devices to meet the needs of different types of crop handling.

In Fig. 4, each robot is equipped with various resources for use in the factory. When a control request is initiated, the ONATS algorithm is employed initially to determine how to manage the request. Subsequently, the relevant robot cars assigned to complete the specific task will receive movement instructions, relocate to the designated position, and carry out the assigned task. During this process, the consumption of relevant resources, such as the remaining liquid in the spraying devices ($c_{m,1}$-$c_{m,3}$), is recorded. In the illustration, three types of liquid resources ($K = 3$) are carried by the robot, with its control unit activating the nozzle to spray different liquids onto the lettuce to complete related tasks.

By conducting experiments in this micro-environment, we can more clearly analyze the efficiency and effectiveness of the proposed method, which is beneficial for in-depth research on the application prospects of adapters in modular plant factories. Besides this, we've also conducted a series of experiments with

[3]https://www.raspberrypi.com/

$$\Omega(\boldsymbol{x}^{(n)}) = \begin{cases} \sum_{j=1}^{J} \left(\mathcal{L}(\beta_j - w_j^{(n)}) + \frac{(\mathcal{U}-\mathcal{L})(\exp(\frac{\alpha}{c_j}(w_j^{(n)}+x_j^{(i)}))}{\exp(\alpha)-\exp(\frac{\alpha}{\alpha-1})} \frac{c_j}{\alpha} - \exp(\frac{\alpha}{c_j}\beta_j) + \frac{\mathcal{L}}{\alpha}(w_j^{(n)} + x_j^{(i)} - \beta_j) \right) & , w_j^{(n)} \in [0, \beta_j] \\ \sum_{j=1}^{J} \left(\frac{(\mathcal{U}-\mathcal{L})(\exp(\frac{\alpha}{c_j}(w_j^{(n)}+x_j^{(i)}))}{\exp(\alpha)-\exp(\frac{\alpha}{\alpha-1})} \frac{c_j}{\alpha} - \exp(\frac{\alpha}{c_j} w_j^{(n)}) + \frac{\mathcal{L}}{\alpha} x_j^{(i)} \right) & , w_j^{(n)} \in [\beta_j, c_j] \end{cases} \tag{17}$$

TABLE II
EXPERIMENTAL PARAMETERS

Item	Value	Item	Value	Item	Value
$\mathcal{N}$	300	J	100	$\mathscr{A}^{(n)}$	$U(1,100)$
M	10	$\delta^{(n)}$	$U(50,200)$	$\mathscr{B}_{m,k}$	$U(1,10)$
K	10	$\eta_{m,k}^{(n)}$	$U(0,c_{m,k})$	$c_{m,k}$	$U(1,300)$

the experimental synthesis data so that the effects of different system settings will be investigated fluently. We also set certain threshold ranges for various parameters in order to acquire random data according to the designed scenario. The basic values of parameters in numeric experiments are shown in Table II.

B. Introduction to Baselines

Prior research has mainly focused on optimizing for generic problem scenarios, without specifically addressing the unique context of modular plant factories. In this paper, we investigate this particular scenario, studying how the factory machines can leverage their distinctive services and resources to help improve crop quality, and find that it is essentially an online multidimensional knapsack problem. So we develop an approximate quantitative model to reveal the relationship between task allocation strategies and final harvest outcomes, and design an α-competition algorithm called ONATS to optimize the overall yield through online task allocation.

Since the ONATS algorithm is the first attempt to solve the request scheduling problem in the plant factory environment, the existing algorithms are not suitable for direct application here because they are not designed for this application. Therefore, we select some similar but representative methods as baselines to compare with the proposed approach ONATS in this paper:

1) *Fixed threshold-based Allocation [24] (*FTA*):* The FTA is also a kind of threshold function-based method, but it differs from the ONATS algorithm because it uses the same threshold function $\phi_{m,k}(w) = \sqrt{\mathcal{U} \times \mathcal{L}}$ for all the adapter-service pairs.
2) *Greedy Resource Allocation [25] (*GRAL*):* This algorithm utilizes a ranking function as its greedy strategy to evaluate the benefits of different resource allocation decisions. It does not take into account current resource utilization, but rather focuses on maximizing the allocation's immediate benefit.
3) *Balanced Resource Allocation [26] (*BARA*):* Inspired by the load balancing strategy introduced in [26], we incorporate it into the BARA algorithm for achieving efficient task scheduling. This method allocates resources according to the remaining space of all "knapsacks"j or $(m,\ k)$. It would like to make the "knapsacks" balanced so that the resources of them can be equally consumed.
4) *Linear Reservation Policy [12] (*LINRP*):* This algorithm uses a reservation strategy. It is designed with the idea of determining the implicit admission cost and admitting the incoming item only if its value is higher than or equal to the admission cost. The admission cost is an increasing function of the current utilization, where higher utilization leads to a higher admission cost.
5) *Exponential Reservation Policy [12] (*EXPRP*):* EXPRP shares design similarities with LINRP, but uses different exponential definitions for the normalized utilization and reservation functions to evaluate the admission cost. Its reservation strategy is based on an exponential knapsack utilization function.

The used code for ONATS is released.[4]

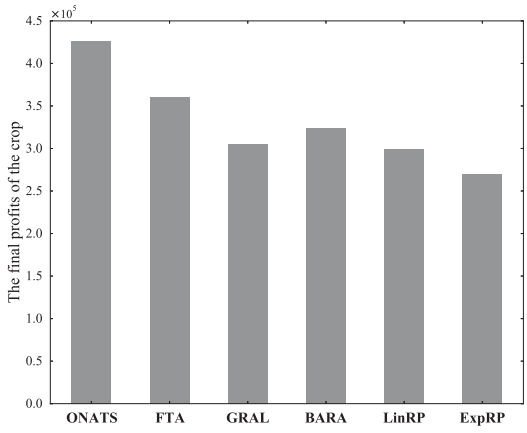

Fig. 5. Profits of different approaches.

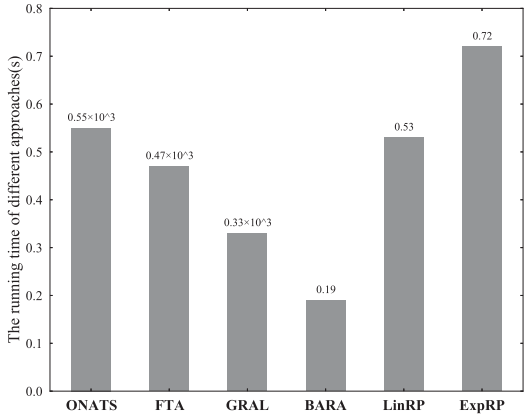

Fig. 6. Running time of different approaches.

C. Comparisons and Analysis

1) Performance Comparison: The comparison results are shown in Fig. 5. With these approaches applied in the same system, it can be found that the proposed ONATS works much better than the others, where the final gain of the crop is 18.2% higher than that of FTA, 39.9% of GRAL, 31.6% of BARA, 42.6% of LINRP and 58.2% of EXPRP.

This result can be attributed to the fact that our method ONATS considers the limited nature of resources and the infinite arrival of task requests in its design, reserving resource space for potentially high-quality future requests, thereby providing greater

[4]https://github.com/xuexizi/ONATS

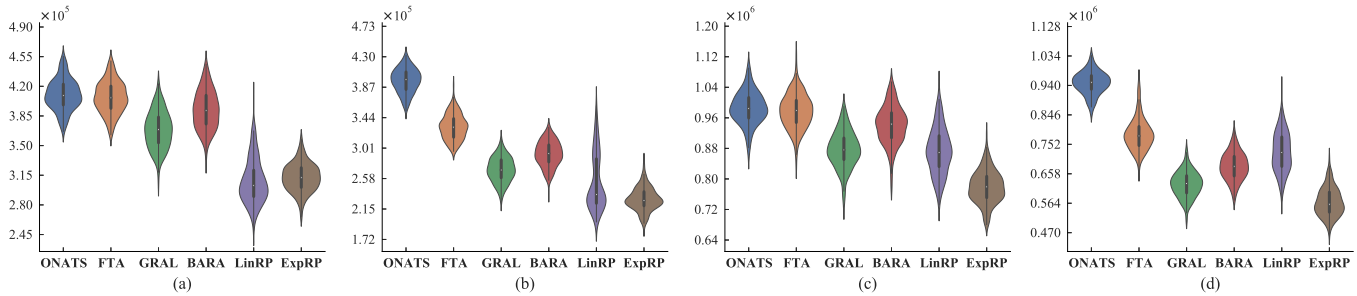

Fig. 7. The impact of the price-efficiency ratio of resources and the ratio of resource supply and demand.

operational flexibility. Additionally, the function thresholds in ONATS are dynamically adjusted as requests arrive and are processed, allowing it to better adapt to the remaining capacity of the knapsack and the changing task demands. As a result, method ONATS can achieve higher gains.

2) Comparison About Running Time: However, achieving better performance comes at a cost. As depicted in Fig. 6, the results of the runtime evaluation demonstrate that FTA executes in approximately 14.5% less time than ONATS. Notably, the BARA strategy has the best average service response time owing to its simplicity, algorithms LINRP and EXPRP also have a relatively short running time, but they both fail to generate a profitable outcome. Therefore, a trade-off between performance and cost must be considered. Fortunately, in plant factories powered by the MEC paradigm, the computing unit or edge server is only accountable for a localized environment, providing sufficient time to handle control requests with ONATS before the subsequent request arises.

3) Comparison About Generality: To evaluate the generality of the approaches, we can first define the *resource sensitivity* in this paper with:

$$RS = \bar{\mathscr{A}} \ln(\bar{\mathscr{B}}) \tag{18}$$

where $\bar{\mathscr{A}}$ is the mean value of $\{\mathscr{A}^{(i)}\}$ and $\bar{\mathscr{B}}$ is the average of $\{\mathscr{B}_{m,k}\}$. The RS index reflects the how the resource will impact in the profit gain. And we can define the *supply-demand intensity:*

$$SDI = \bar{\delta}/\bar{c} \tag{19}$$

where $\bar{c}$ is the mean of $\{c_{m,k}\}$ and $\bar{\delta}$ is the average of $\{\delta^{(i)}\}$. The SDI index shows how the available resources can afford the demands. We can then check the generality of the proposed approach by executing it with various RS and SDI values. To achieve this, we conducted 200 experiments with random requests and depicted the final profits of the six approaches using violin plots. The results are shown in Fig. 7. The results in Fig. 7(b) (RS = 80.47, SDI = 0.625) and Fig. 7(d) (RS = 230.26, SDI = 0.625) indicate that, under conditions of strong supply-demand intensity, i.e., when available resources are adequate, ONATS demonstrates a significantly higher profit than the other methods. Conversely, the comparison of Fig. 7(a/c)(RS = 80.47/230.26, SDI = 0.375) and Fig. 7(d) (RS =, SDI = 0.375) shows that the advantage of ONATS is diminished as the SDI weakens.

4) Comparison About Balancing: We conducted an analysis of the resource utilization over time for the six different approaches to investigate their effectiveness. The data presented in Fig. 8 was used for this purpose. The figure clearly indicates that unlike several other allocation methods, algorithms ONATS, LINRP, and EXPRP ensure that there is always available capacity for the next request. They consider the current resource utilization and reserve a certain amount of resources for future requests, rather than completely filling up the capacity. However, as seen in Fig. 5, the gains obtained by the LINRP and EXPRP algorithms are not ideal because their models are designed for scenarios where requests are much smaller than supply. Therefore, the ONATS algorithm is more suited to the scenario of plant factories.

D. Impacts of System Configurations

The above comparison shows that ONATS is practical. In addition, we would like to discuss the impact of system configurations to see how they will affect the results.

1) The Impact of Utility Function γ: In Fig. 9, when considering only the increase of $\bar{\mathscr{A}}$, it is evident that the profit obtained by FTA surpasses ONATS when the value of $\bar{\mathscr{A}}$ is small. This observation leads us to go deeper in investigating not only the average but also the range of $\{\mathscr{A}\}$.

By denoting $\mathscr{A}_{\max}/\mathscr{A}_{\min}$ as the range ratio, we maintain all other parameters constant and gradually increase the range ratio from 20 to 50. The results are shown in Fig. 10, where we conducted 30 experiments per group to eliminate interference caused by extreme cases. From these results, we can conclude that at the current system benchmark value, ONATS is comparable to the FTA algorithm when the range ratio is between 25 and 30. However, when it exceeds this value, ONATS outperforms FTA.

When we turn to Fig. 11 where only the increase of $\bar{\mathscr{B}}$ is considered, it is clear that the profit obtained by ONATS dominates those of other baselines.

2) Impact of Resource Requirement: We cannot make bricks without straw — without sufficient resources, the system's performance may decline, leading to lower-quality outcomes, as illustrated in Fig. 12. The performance of the approaches can be categorized into three tiers based on the figure. ONATS excels in the first tier, surpassing the other methods. The FTA and LINRP methods form the second tier, exhibiting approximately a

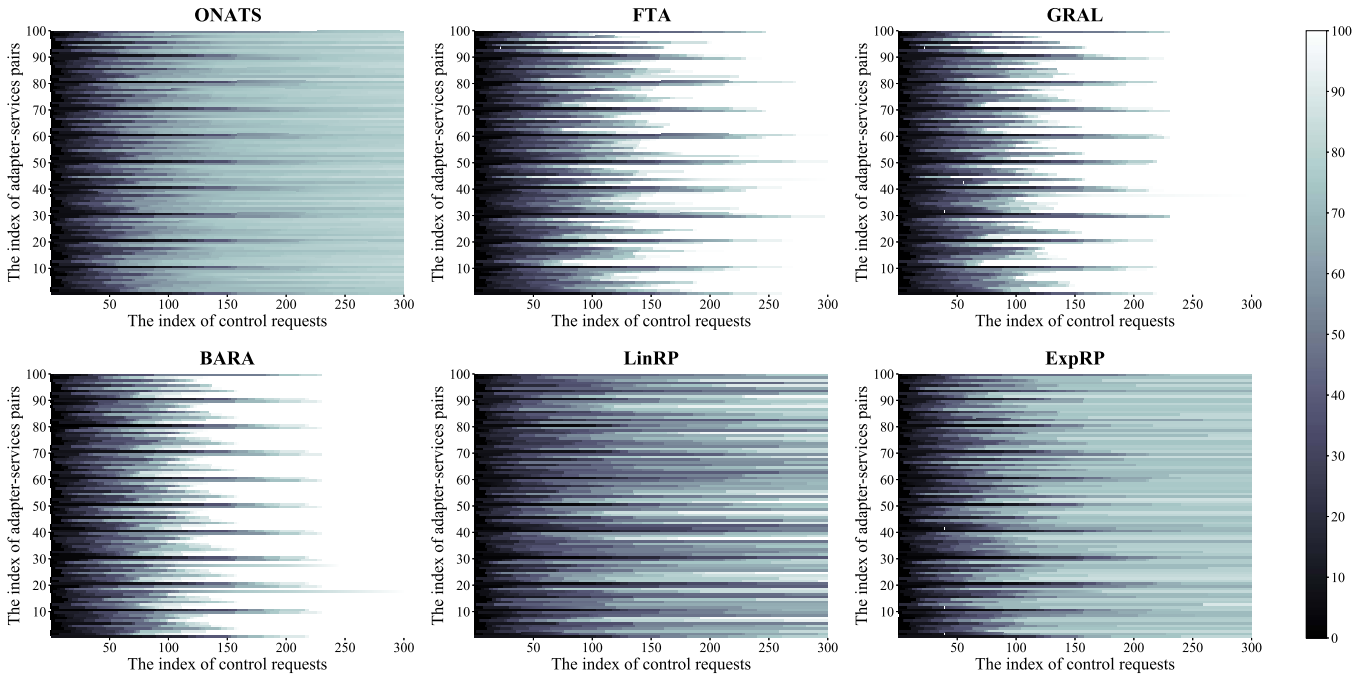

Fig. 8. Resource utilization varies with the coming of control requests.

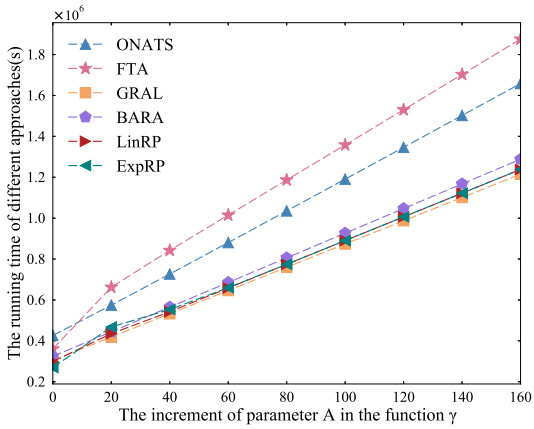

Fig. 9. Profit varies with parameters $\bar{\mathscr{A}}$.

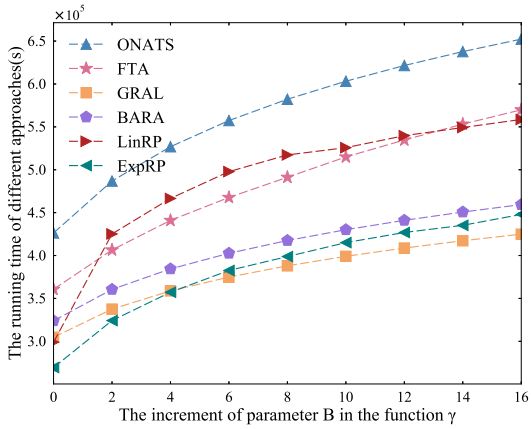

Fig. 11. Profits varies with parameters $\bar{\mathscr{B}}$.

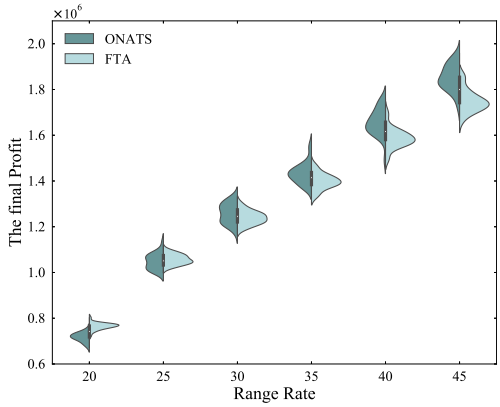

Fig. 10. Effect of the range ratio of $\mathscr{A}$.

25% performance gap compared to ONATS , which widens with increasing δ values. Furthermore, even with the same increase in requested resources, the ONATS method demonstrates a slower decline than the other methods. This advantage of ONATS stems from its proactive nature and enhanced service provision capabilities.

3) Impact of Problem Scale: To assess the impact of system scale, we maintain a consistent system configuration and vary the number of adapters and service types separately and record the running time of processing requests. Fig. 13 illustrates the effect on the running time of the ONATS algorithm as the problem scale changes. Our findings indicate that the execution time increases linearly as the number of adapters increases, providing

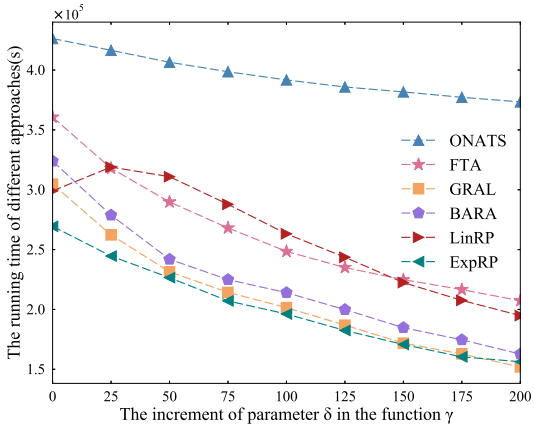

Fig. 12. Profits varies with parameters $\bar{\delta}$.

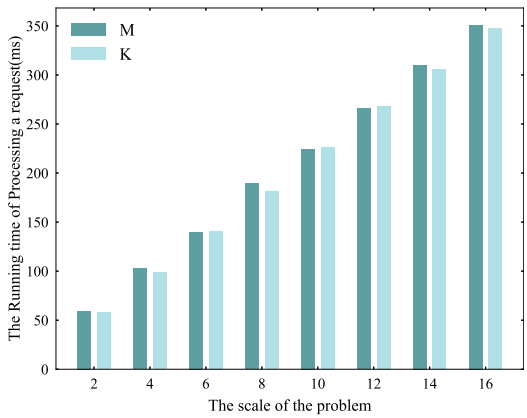

Fig. 13. The impact of the number of bots on execution time.

evidence that the algorithm's complexity is consistent across iterations. Additionally, holding the number of adapters constant and increasing the service type will also result in the same linear result.

VII. Conclusion and Future Work

In this paper, we investigate resource allocation and task scheduling issues in modularized plant factories' service provisioning, and model adapters' behavior and their corresponding contributions as an online decision-making problem. To address the problem, we propose an online approach, called ONATS, that dynamically adjusts task allocation and scheduling strategies based on current online multi-knapsack research. By doing so, the factory can maximize profits while providing optimal care for the crops. Our comparative results demonstrate that this algorithm outperforms other baseline methods when the relationship between resource supply and demand is appropriate.

In our future work, we plan to tackle the challenge of executing different methods for multiple plants under the same task requirements. Additionally, we aim to further optimize this algorithm using current AI algorithms to achieve higher efficiency and a wider range of application scenarios.

Acknowledgments

Zhengzhe Xiang analyzed the problem, designed the ONATS and wrote the paper. Xizi Xue and Zhengzhe Xiang conducted the experiments. Minyi Guo and Schahram Dustdar joined the discussion of modeling, and contribute to the formulation of the problem. Yuanyi Chen proofread the paper and provided hardware.

References

[1] F. Kalantari, O. M. Tahir, R. A. Joni, and E. Fatemi, "Opportunities and challenges in sustainability of vertical farming: A review," *J. Landscape Ecol.*, vol. 11, no. 1, pp. 35–60, 2018.

[2] T. Kozai, G. Niu, and M. Takagaki, *Plant Factory: An Indoor Vertical Farming System for Efficient Quality Food Production*. New York, NY, USA: Academic, 2019.

[3] Z. Tian, W. Ma, Q. Yang, and F. Duan, "Application status and challenges of machine vision in plant factory—A review," *Inf. Process. Agriculture*, vol. 9, pp. 195–211, 2022.

[4] S. Santiteerakul, A. Sopadang, K. Yaibuathet Tippayawong, and K. Tamvimol, "The role of smart technology in sustainable agriculture: A case study of Wangree plant factory," *Sustainability*, vol. 12, no. 11, 2020, Art. no. 4640.

[5] S. Deng et al., "Dependent function embedding for distributed serverless edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 10, pp. 2346–2357, Oct. 2022.

[6] S. Deng, C. Zhang, C. Li, J. Yin, S. Dustdar, and A. Y. Zomaya, "Burst load evacuation based on dispatching and scheduling in distributed edge networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 8, pp. 1918–1932, Aug. 2021.

[7] H. Zhao, S. Deng, Z. Liu, J. Yin, and S. Dustdar, "Distributed redundant placement for microservice-based applications at the edge," *IEEE Trans. Serv. Comput.*, vol. 15, no. 3, pp. 1732–1745, May/Jun. 2022.

[8] P. Cruz, N. Achir, and A. C. Viana, "On the edge of the deployment: A survey on multi-access edge computing," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–34, 2022.

[9] B. Cao, Z. Sun, J. Zhang, and Y. Gu, "Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3832–3840, Jun. 2021.

[10] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in UAV-enabled mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3147–3159, Apr. 2020.

[11] V. Priya, C. S. Kumar, and R. Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning," *Appl. Soft Comput.*, vol. 76, pp. 416–424, 2019.

[12] L. Yang, A. Zeynali, M. H. Hajiesmaili, R. K. Sitaraman, and D. Towsley, "Competitive algorithms for online multidimensional knapsack problems," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 3, pp. 1–30, 2021.

[13] H. Halabian, "Distributed resource allocation optimization in 5G virtualized networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 627–642, Mar. 2019.

[14] T. F. Rahman, V. Pilloni, and L. Atzori, "Application task allocation in cognitive IoT: A reward-driven game theoretical approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 12, pp. 5571–5583, Dec. 2019.

[15] E. A. Khalil, S. Ozdemir, and A. A. Bara'a, "A new task allocation protocol for extending stability and operational periods in Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7225–7231, Aug. 2019.

[16] K. Gai and M. Qiu, "Optimal resource allocation using reinforcement learning for IoT content-centric services," *Appl. Soft Comput.*, vol. 70, pp. 12–21, 2018.

[17] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 361–371, 2020.

[18] Y. Huang et al., "Deep adversarial imitation reinforcement learning for QoS-aware cloud job scheduling," *IEEE Syst. J.*, vol. 16, no. 3, pp. 4232–4242, Sep. 2022.

[19] T. J. Givnish, R. A. Montgomery, and G. Goldstein, "Adaptive radiation of photosynthetic physiology in the Hawaiian lobeliads: Light regimes, static light responses, and whole-plant compensation points," *Amer. J. Botany*, vol. 91, no. 2, pp. 228–246, 2004.

[20] S. Im, R. Kumar, M. Montazer Qaem, and M. Purohit, "Online knapsack with frequency predictions," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 2733–2743.
[21] Z. Zhang, Z. Li, and C. Wu, "Optimal posted prices for online cloud resource allocation," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 1, pp. 1–26, 2017.
[22] Y. Zhou, D. Chakrabarty, and R. Lukose, "Budget constrained bidding in keyword auctions and online knapsack problems," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 1243–1244.
[23] B. Sun, A. Zeynali, T. Li, M. Hajiesmaili, A. Wierman, and D. H. Tsang, "Competitive algorithms for the online multiple knapsack problem with application to electric vehicle charging," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 3, pp. 1–32, 2020.
[24] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin, "Optimal search and one-way trading online algorithms," *Algorithmica*, vol. 30, pp. 101–139, 2001.
[25] N. Ferdosian, M. Othman, B. M. Ali, and K. Y. Lun, "Greedy–knapsack algorithm for optimal downlink resource allocation in LTE networks," *Wireless Netw.*, vol. 22, pp. 1427–1440, 2016.
[26] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.

Zhengzhe Xiang received the BS and PhD degrees in computer science and technology from Zhejiang University, Hangzhou, China. He was a visiting scholar with Shanghai Jiao Tong University, Shanghai, China, in 2022. He is currently an associate professor with Hangzhou City University, Hangzhou, China. His research interests lie in the fields of service computing and edge computing. He serves for several international journal like the *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Services Computing*, *IET Communications*, *Digital Communications and Networks* as reviewers. He also serves as PC members of many international conferences.

Xizi Xue is currently working toward the MS degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. Her research interests include Internet of Things technology, edge computing, service computing, and reinforcement learning.

Yuanyi Chen (Member, IEEE) received the BSc degree from Sichuan University, Chengdu, China, in 2010, the MSc degree from Zhejiang University, Hangzhou, China, in 2013, and the PhD degree from Shanghai Jiao Tong University, Shanghai, China, in 2017. He is currently a full professor with the Department of Computer Science and Computing, Hangzhou City University, Hangzhou, China. He has published more than 30 technical papers in major international journals and conference proceedings. His research interests include the Internet of Things and edge computing.

Schahram Dustdar (Fellow, IEEE) is a full professor of computer science and heads TU Wien's Distributed Systems Group. His research interests include distributed systems, edge Intelligence, complex and autonomic software systems. He is the editor in chief of the *Computing*; associate editor of the *ACM Transactions on the Web*, *ACM Transactions on Internet Technology*, *IEEE Transactions on Cloud Computing*, and *IEEE Transactions on Services Computing*. He's also on the editorial boards of the *IEEE Internet Computing* and *IEEE Computer*. He has received the ACM Distinguished Scientist award and Distinguished Speaker Award and the IBM Faculty Award. He is an elected member of Academia Europaea, where he's was Informatics Section chairman from 2015 to 2022. He is an AIIA fellow where he is the current president.

Minyi Guo (Fellow, IEEE) received the PhD degree in computer science from the University of Tsukuba, Japan. He is currently Zhiyuan chair professor and head with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His present research interests include parallel/distributed computing, compiler optimizations, embedded systems, pervasive computing, Big Data, and cloud computing. He is currently on the editorial board of the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, and *Journal of Parallel and Distributed Computing*. He is a fellow of the CCF.