The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Delivering the optimum Sustainable Automated Indoor Vertical Farming

by

Zhiying Chen

Dissertation Presented for the Degree of
MSc in Operational Research with Data Science

August 2022

Supervised by
Dr. Jessica Reis Kitamura and Dr. Julian Hall

Executive Summary

With the explosive growth of the world's population, the demand for food is increasing every day. It is for this reason that a new type of automated indoor farming has been born, which is called vertical farming. This thesis provides a solution on how to run a vertical farm in a fully automated way. Two mixed-integer linear programming (MIP) models, with two speed-up heuristics, are built to solve two optimization problems. One MIP model is to find a placement schedule for the products in a farm of a fixed number of racks in a given month. A greedy algorithm is proposed to speed up the process significantly and it reduces the objective by at most 8%. The other MIP model is to solve the daily robot arrangement problem so as to find the minimum number of robots needed for serving all racks each day. This problem is a typical bin packing problem so Best Fit Decreasing (BFD) heuristic is adopted to provide a highly approximate result of the model efficiently. The results show the effectiveness of the proposed models and algorithms and how they can be applied for picking the optimal farm size and operating an automated vertical farm remotely.

Acknowledgments

It has not been easy to complete my master's studies and accomplish my dissertation, and I would like to thank all those who have helped me along the way.

Firstly, I have been honoured to to start my thesis on this topic provided by Sopra Steria.

I would like to express my sincere appreciation to Dr. Jessica Reis Kitamura and Dr. Julian Hall for their instructions. Dr. Jessica Reis Kitamura is a very considerate supervisor. She had a meeting with me patiently every week to check how my thesis was progressing and offered me much support. Dr. Julian Hall is my academic supervisor and also my personal tutor during my master's year. Julian is a very nice person who cared about his students and he gave me guidance on my writing during my dissertation.

I am also grateful to Mr. Kaloyan Bukovski from Sopra Steria, who is also an alumnus of the University of Edinburgh. He put much effort into organizing the joint writing between the company and the university, thus making the dissertation writing experience very good. I would also like to thank Mr. Andreas Francois Vermeulen for explaining some details of this problem to me.

I am thankful to all lecturers from the School of Mathematics at the University of Edinburgh for their conscientiousness and professional teaching approaches, which made me love operational research very much and decide to devote myself to OR-related work in the future. In particular, Prof. Jacek Gondzio, with his great expertise and humorous way of delivering lectures, is able to make complex and difficult knowledge come alive and interesting, which triggered me to explore more about the OR field.

Finally, I want to thank my family and friends for their accompany and support during this year. I thank my parents for their willingness to give me unconditional support even in very difficult situations so that I am able to complete my studies successfully. I would like to thank my flatmates Chenyun, Zijing, and Xiaogang for being there for me this year and for surprising me from time to time so that I do not feel lonely and helpless in a foreign country.

University of Edinburgh – Own Work Declaration

This sheet must be filled in, signed and dated - your work will not be marked unless this is done.

Name: Zhiying Chen
Matriculation Number: s2140034
Title of work: Delivering the optimum Sustainable Automated Indoor Vertical Farming

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Not sought or used the help of any external professional academic agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course handbook

I understand that any false claim for this work will be penalised in accordance with the University regulations (`https://teaching.maths.ed.ac.uk/main/msc-students/msc-programmes/statistics/data-science/assessment/academic-misconduct`).

Signature: .....

Date: April 13, 2023

Contents

List of Tables

List of Figures

1 Introduction

The total size of the world population is likely to increase from its current 7 billion to 8–10 billion by 2050. [28] However, at constant densities, the world's urban land cover will only double between 2000 and 2050 as the world population doubles. [9] The United Nations anticipates that food demand will have to rise by 70% to sustain this growing, increasingly urbanized population.[24] Therefore, the task of supplying adequate food for everyone in a sustainable, efficient, and cost-effective manner is becoming increasingly important in urban cities. The yields of traditional farming were constrained by many factors, such as severe climate change, unexpected pests, transportation obstacles, and so on. [17] A new farming concept, Urban Smart Vertical Farming (USVF) appeared with the hope to remove these constraints.

USVF is a system for securing food production that may be implemented vertically in any adaptive reuse, retrofit, or new building. [34]USVF has the ability to grow crops all year, with more control over food safety and bio-security, while reducing inputs like water, pesticides, herbicides, and fertilizers. [7]. As the growing process is indoors, the indoor facilities provide the most suitable conditions for the plants to grow, such as light, temperature, air concentration, nutrient solution, etc. At the same time, thanks to the rapid spread of automation, the planting process including watering and lighting can be completed by robots as well. Plant growth may be observed via probes, and the robots can sense whether the plants are growing well by means of artificial intelligence such as computer vision to take better care of them. [32] This significantly reduces labor costs.

Vertical farms are not without their drawbacks, with detractors pointing out that it is more energy-consuming to sustain such sophisticated ecosystems than traditional farming. To solve this problem, several vertical farms use sustainable energy sources and recycle a large portion of their waste thank to the fact that they are large enough to be equipped with solar panels. In addition, the adoption of energy-efficient LED lighting minimizes power consumption, and blue and red-light shades are even more cost-effective to operate. [16] In fact, the concept of vertical farming has been successfully implemented in a number of developed countries, such as AeroFarms in the United States and Sky Greens in Singapore. [32]

The rest of the paper is organized as follows. Section 2 gives a brief background of the addressed problem and then summarizes some optimization methodology from previous literature. Some prerequisite knowledge mentioned in the thesis is also introduced in this part. Section 3 describes how MIP (mixed integer linear programming) models can be used to plan a vertical farm so that it can be as profitable as possible. Section 4 presents the results and conducts a discussion. Section 5 concludes the work and gives future suggestions.

2 Background

This section introduces the background of the thesis and it is divided into the following three parts. Firstly, the problem background and research questions are introduced. Next, some optimization methods of planning vertical farming are summarized from the previous literature. Lastly, some essential knowledge included in the thesis paper is presented within its right context. The goal of this information is to build a foundation of ideas to make it easier to understand the task.

2.1 Problem formulation

This is an industrial problem. There are local demands and overseas demands for 105 agricultural products each month for the past 9 years. The task is to prepare a scheduling solution for a series of 24/7/365 vertical indoor farms for the coming year. The data set is synthetically generated and provided by *Sopra Steria*.

Each product can be produced on a shelf of a farm with its own parameters. An example is given in Table 1.

Product	Amount/Shelf $[kg/m^2]$	Light $[lux]$	Water $[L/day]$	Water Frequency $[times/day]$	Growth Cycle $[days]$	Retail Price $[pounds/shelf]$
cabbage	7	16800	4.68	7	33	4.83
lettuce	6	9000	5.27	7	13	3.3

Table 1: An example of plant growth information

A typical farm consists of a set of N^2 racks on a square farm in N rows with N racks in each row, of which the area is $(2N + 3)^2$ square meters. All racks include 10 (1-meter square) shelves. They are configured in a grid with a meter gap between the racks. The rent for a farm is 200 pounds per square meter per year, and each rack costs 10 pounds.

A schematic diagram of a farm is shown in Fig. 1.

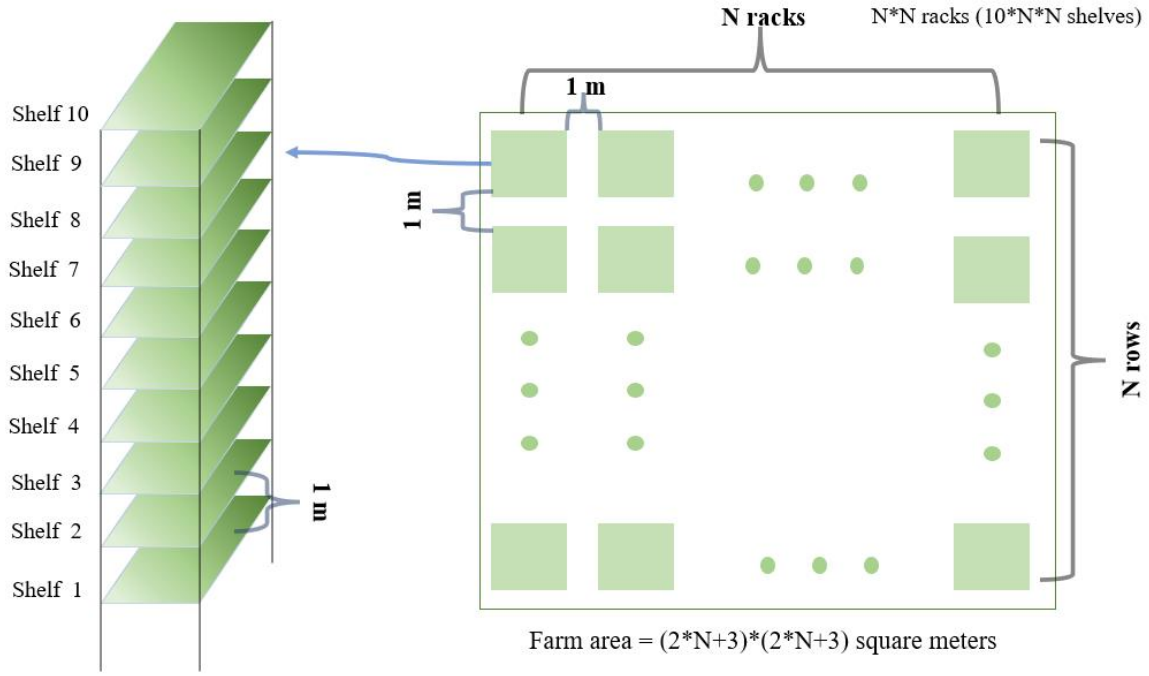


Figure 1: Schematic diagram of a typical farm

There are two groups of robots taking care of the complete farm. They can move at 0.5 meters per minute, so 2 minutes to move from one rack to another. Robots can only move between racks when on the ground. They can climb a rack at 0.2 meters per minute (up/down speed), so 5 minutes to

move from one shelf to another. Every time they finish a climbing round, they need to go back down to the bottom to start another round or move to the next rack. Each robot costs 420 pounds to rent every month (30 days).

One group of robots that supply water to grow the plants is named the water robot. They take 90 seconds to water a shelf with one litre of water. Every time they work for two hours, they need 45 minutes to be charged (to rest). It can be assumed that they work 8 times per day, with 2 hours each. The total working time of a water robot is 16 hours per day, and the rest of the day is left for it to move to charging stations and be charged.

The other group of robots supplying light to grow the plants is called the light robot. They supply 2500 lux/minute. Every time they get recharged for 3 hours; they need 75 minutes to be charged (to rest). It can be assumed that they work 5 times per day, with 3 hours each. The total working time of a light robot is 15 hours per day, and the rest of the day is left for it to move to charging stations and be charged.

Fig. 2 shows how robots move both horizontally and vertically. It can be discovered that it takes a long time for robots to go up and down, and therefore one requirement is to put products that require less frequency of water on the top, and vice-versa.

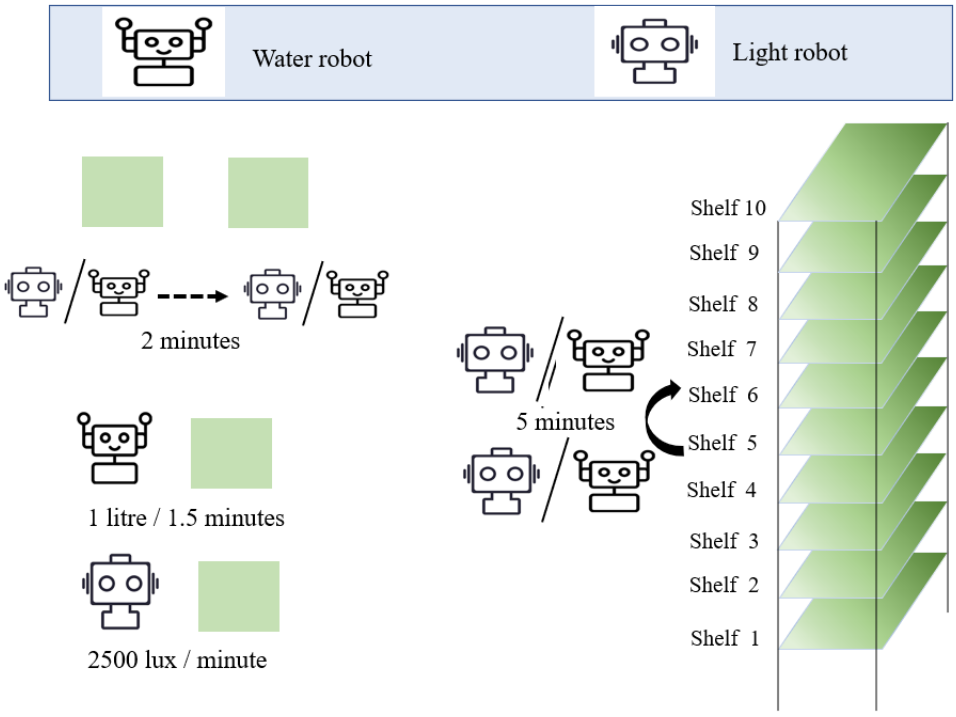


Figure 2: Schematic diagram of robot movements

The profit is calculated as the income minus the cost. The income only comes from the sold products, while the cost includes two parts. One is the setup cost of the farm and racks for the whole year. The other is the cost of renting robots each month. It is assumed that the water and light costs are included in the robot cost.

2.1.1 Research questions

The main goal of this research is to maximize the total profit by choosing an appropriate farm size, of which selected products can be planted in its racks. To address the described problems, some initial research questions have been explored. The first question is which products are worth being planted on this vertical farm. The second question is what the predicted demands for all products for the next year are. Since the profit consists of the product income, the farm cost, and the robot cost, where the former needs to be maximized and the latter two minimized. To maximize the profit, the third question is what an appropriate farm is and how to arrange products in its racks.

2.1.2 Data pre-processing

There are over twenty different types of each product available on the market, and most of them meet the requirements of the demand. An example is shown in Table 2.

Product Type	Amount/Shelf $[kg/m^2]$	Light $[lux]$	Water Amount $[L/day]$	Water Frequency $[times/day]$	Growth Cycle $[days]$
Type-Uniform	7	16800	4.68	7	33
Type-Tango	2	1560	3.26	9	19

Table 2: Different types of cabbages

There are some types in which the cycle length is not reasonable for a product to grow maturely. In this case, it is necessary to select one type to produce on the farm. To exclude these situations, the cycle length distributions of all products are checked to be normal distribution by Kolmogorov–Smirnov test [4] in the first place because statistically, the Kolmogorov–Smirnov test (K-S test) is a nonparametric test of the equality of continuous, one-dimensional probability distributions that can be used to compare the differences of a sample with a reference probability distribution, or to tell whether two samples are of the same distribution or not. [4] Only those with cycle lengths in the 95% confidence intervals of their distributions are considered.
Given the notations below.

- $\tilde{P}$: the set of all products
- $p \in \tilde{P}$: product p
- T_p: the set of all types of product $p \in \tilde{P}$
- $t \in T_p$: type t of product p
- $c_{p,t}$: the cycle length of type $t \in T_p$
- $w_{p,t}$: the total amount of water needed for 1 kilogram of type $t \in T_p$ to grow maturely.
- $l_{p,t}$: the total light needed for 1 kilogram of type $t \in T_p$ to grow maturely.
- $f_{p,t}$: the water frequency of type $t \in T_p$ per day.

For type t_0 of product $p \in \tilde{P}$, $w_{p,t}$ is projected to be normalized $sw_{p,t} \in [1, 2]$ by

$$sw_{p,t_0} = \frac{w_{p,t_0} - \min\limits_{t \in T_p} w_{p,t}}{\max\limits_{t \in T_p} w_{p,t} - \min\limits_{t \in T_p} w_{p,t}} + 1$$

The same is done for $l_{p,t}$ to be standardized $sl_{p,t} \in [1, 2]$.

$$sl_{p,t_0} = \frac{l_{p,t_0} - \min\limits_{t \in T_p} l_{p,t}}{\max\limits_{t \in T_p} l_{p,t} - \min\limits_{t \in T_p} l_{p,t}} + 1$$

Then the Price Performance Ratio (PPR) is defined as

$$PPR_{p,t} = c_{p,t} \times f_{p,t} \times sw_{p,t} \times sl_{p,t}$$

Type $t_p = \underset{t \in T_p}{argmin}\, PPR_{p,t}$ is picked for each product $p \in \tilde{P}$.

Only one type of each product is kept after selection. The samples are reduced to 105 products $\times$ 1 type from 105 products $\times$ 27 types.

2.2 Literature review

In recent years, with the rise of vertical farms, there has been a growing literature on the use of operations research to address vertical farming planning. Back in 2012, Yang et al.[42] researched the selection and arrangement of crop combinations to be grown in a plant factory. The authors propose a crop scheduling methods which takes into account the factor of crop price fluctuation and contracts between plant factories and retailers. Crop scheduling is formulated as a mixed integer programming (MIP) problem and the objective is to maximize the revenue of a plant factory by crop scheduling. It is solved using branch-and-bound algorithms. For plant factory operation and administration, the solution offers guidelines for crop selection and planting timing.

Yang et al. [43] provided a further discussion on the previous questions. The plant factory scheduling problem was formulated as a mixed integer linear programming (MIP) problem. The objective function is also to maximize revenue by considering several practical operating conditions such as (1) crop market value per unit and time, (2) crop adjacency issue, (3) cleaning and maintenance, and (4) sunlight blocking by a solar pan. The authors proposed a heuristic algorithm, named Heuristic Plant Factory Scheduler (HPFS), which applies a recursive technique to schedule crops rack by rack. Compared to solvers, HPFS dramatically reduces run time as the number of racks increases, however, scarifies about only 15% of revenue.

Belarmino and Gabriel[6] studied a vertical farming problem from different perspectives, considering new forms of customer-producer relationship, involving long-term cooperation agreements where the product volumes are agreed upon and the demand is guaranteed in advance. They introduced a lexicographic bi-objective optimization approach that, in addition to cost minimization, aims at minimizing the risk of not meeting the agreed demands. Several experiments showed the high complexity of the solution. The paper provides a new solution procedure to the new type of relationships in the food logistics chain.

More recently, Maxence and Alberto[16] studied the Vertical Farming Elevator Energy Minimisation Problem (VFEEMP), where the goal is to decide the tasks should be completed in which sequence and to reduce the distance traveled by elevators so as to reduce energy consumption. It presented three Mixed-Integer Linear Programming (MIP) formulations with valid inequalities, as well as a Constraint Programming model, after showing that the decision problem associated with the VFEEMP is NP-complete.

There is also a lot of literature applying genetic algorithms to solve the job scheduling problems of vertical farming.[31, 37] A master thesis[5] applied the genetic algorithm to vertical farming, which transformed the vertical farming problem into famous Job Shop Scheduling (JSP) problems by monitoring the vertical farm and scheduling the farming activities in order to schedule the harvests. The main objective is to find a solution to reduce the makespan of the farming operation as much as possible while satisfying the constraints of processors' maximum ports and loads. Genetic algorithms need to generate feasible solutions, which is difficult to guarantee, through crossover and mutations. To overcome this challenge, this article innovatively designed an adaptive penalty function where the penalty of a feasible solution is zero.

The above articles only analyzed how MIP can bring possible solutions to one particular problem of vertical farms, but did not put forward suggestions for the overall planning of vertical farms. In fact, in the operation of vertical farms, each step is closely connected to the other. Consequently, this paper gives an example of how one can plan each step and connect them, especially how to relate two or more MIP models with appropriate logic to provide a systematic solution for vertical farming.

2.3 Prerequisite knowledge

This part will cover some of key information mentioned in this paper.

2.3.1 Stationarity

A stationary time series is one whose properties do not depend on the time at which the series is observed.[41]

Def 2.1 (Stationary process)**.** Let $\{X_t\}$ be a stochastic process and it is **(weakly) stationary** if

(i). $\mu_\chi(t)$ is independent of t.

(ii). $\gamma_\chi(t+h,t)$ is independent of t for each h.

where the second condition means that the auto-covariance $\gamma_\chi(r,s)$ has nothing to do with either r or s, but is determined by the time interval $|r-s|$.

The Augmented Dickey-Fuller test can be used to check whether a time series is stationary or not by testing a unit root in a univariate process in the presence of serial correlation. [13]

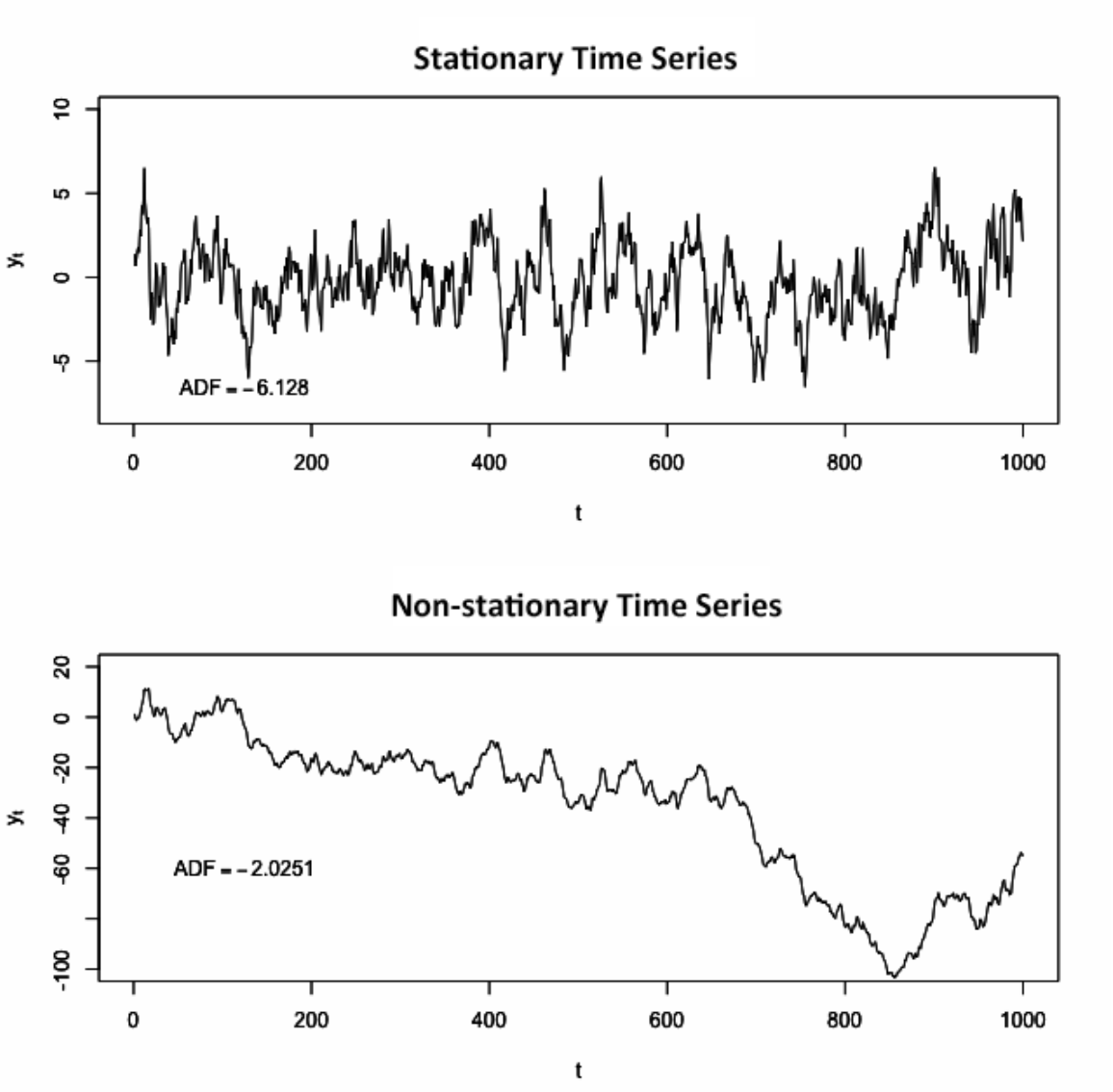


Figure 3: Comparison of stationary and non-stationary time series[38]

Two simulated time series processes, one stationary and the other non-stationary are shown above in Fig.3. The augmented Dickey-Fuller (ADF) test statistic is reported for each process; non-stationarity cannot be rejected for the second process at a 5% significance level.

2.3.2 White noise process

Def 2.2 (White noise process)**.** A time process $\{X_t\}$ is called a white noise process if a sequence of uncorrelated random variables, each with a constant mean $E(X_t) = \mu_X$, usually assumed to be 0, and constant variance $Var(X_t) = \sigma_X^2$. It is the simplest but a very important stationary time series.

Generally speaking, a white noise process is a random process of uncorrelated random variables, that has a mean of zero and a finite variance. If a process is considered to be white noise, then it is unpredictable due to the randomness property.[41]

The Ljung–Box test is a type of statistical test of whether a time series is a white noise process, which is usually used to test residuals. It is based on the auto-correlation plot. However, instead of testing randomness at each distinct lag, it tests the "overall" randomness based on a number of lags.[12]

In a time series, a *lag* is a fixed amount of passing time. One set of observations in a time series is plotted (lagged) against a second, later set of data. The k^{th} lag is the time period that happened k time points before time i. For example, $Lag_1(Y_2) = Y_1$ and $Lag_4(Y_9) = Y_5$.

The autocorrelation function (ACF) [33] of a time series $\{u_t\}$ at lag s is defined as

$$\rho_s = \frac{E(u_t, u_{t-s})}{\sqrt{Var(u_t)Var(u_{t-s})}}; \qquad s = 0, \pm 1, \pm 2, \cdots$$

The Ljung–Box test is adopted to test whether the local demand time series of each product are white noise or not in Section 3.2.1.

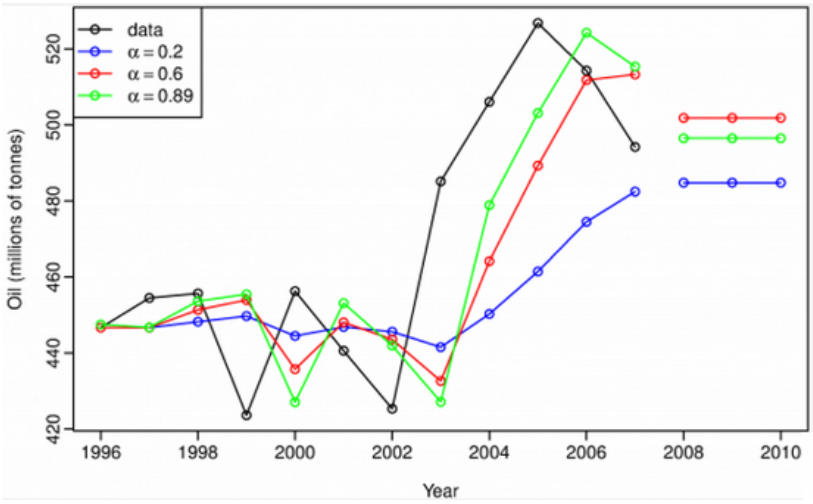
2.3.3 Exponential Smoothing

Exponential smoothing refers to the use of an exponentially weighted moving average (EWMA) to "smooth" a time series. [11]

Def 2.3 (Simple Exponential Smoothing (SES))**.** Let $\{y_t\}$ be a time series process, then Simple Exponential Smoothing predicts the time $T+1$ as follows.

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots, \tag{7.1}$$

where $0 \leq \alpha \leq 1$ is the smoothing parameter. The one-step-ahead forecast for time $T+1$ is a weighted average of all of the observations in the series $y_1, \ldots, y_T$. The rate at which the weights decrease is controlled by the parameter α.



Forecasts		
$\hat{y}_{T+h\|T}$		
484.8	501.8	496.5
484.8	501.8	496.5
484.8	501.8	496.5

Figure 4: Simple exponential smoothing applied to oil production in Saudi Arabia (1996–2007)[1]

The forecasts for oil production in Saudi Arabia [23] of the period 2008–2010 are plotted in Fig.4, where it is smooth at first but as for longer range forecasts, SES assumes that the forecast function is "flat", that is $\hat{y}_{t+h} = \hat{y}_{t+1}$.

Def 2.4 (Holt-Winters' additive method)**.** Let $\{y_t\}$ be a time series process, then the component form

for the additive method is:

$$\begin{aligned}\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)}\\ \ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})\\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}\\ s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m},\end{aligned}$$

where k is the integer part of $(h-1)/m$, which ensures that the estimates of the seasonal indices used for forecasting come from the final year of the sample. The level equation shows a weighted average between the seasonally adjusted observation $(y_t - s_{t-m})$ and the non-seasonal forecast $(\ell_{t-1} + b_{t-1})$ for time t. The trend equation is identical to Holt's linear method. The seasonal equation shows a weighted average between the current seasonal index, $(y_t - \ell_{t-1} - b_{t-1})$ and the seasonal index of the same season last year (i.e., m time periods ago).

Def 2.5 (Holt-Winters' multiplicative method)**.** The component form for the multiplicative method is:

$$\begin{aligned}\hat{y}_{t+h|t} &= (\ell_t + hb_t)s_{t+h-m(k+1)}\\ \ell_t &= \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1})\\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}\\ s_t &= \gamma\frac{y_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma)s_{t-m}\end{aligned}$$

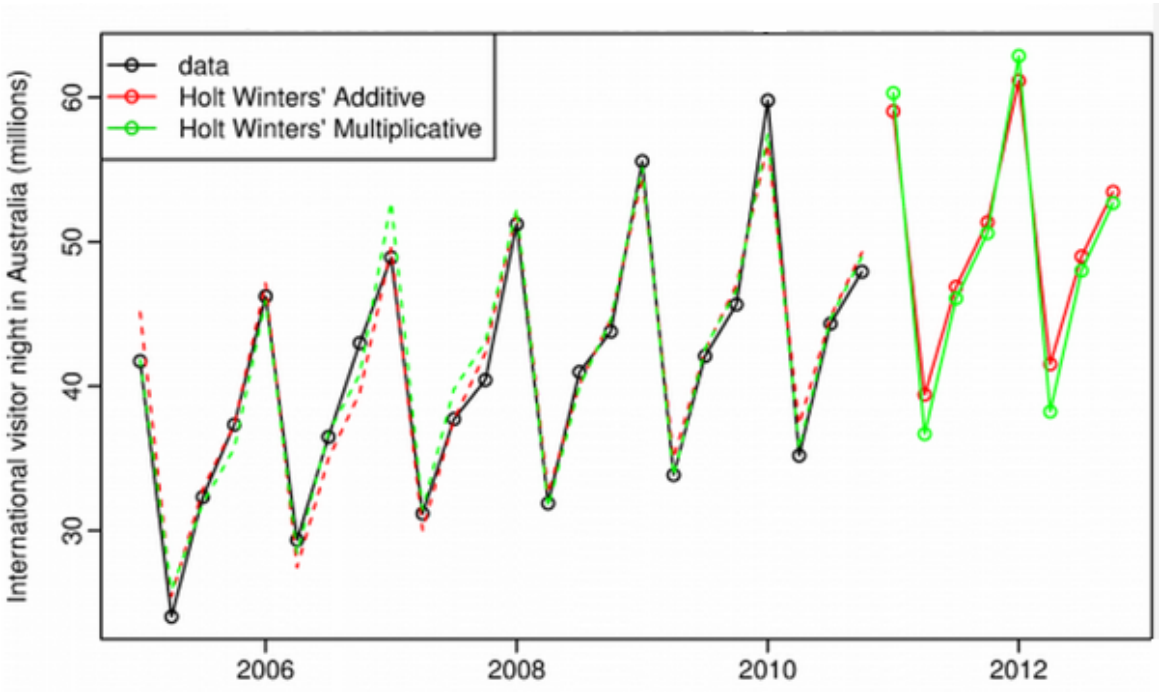


Figure 5: Forecast international visitor nights in Australia using the Holt-Winters method[1]

Holt-Winters' method with both additive and multiplicative seasonality is applied to forecast quarterly visitor nights in Australia spent by international tourists. Fig.5 shows the data from 2005, and the forecasts for 2011–2012. The data show an obvious seasonal pattern, with peaks observed in the March quarter of each year, corresponding to the Australian summer.

Holt-Winters Exponential Smoothing is used for forecasting time series data that exhibits both a trend and a seasonal variation.[18] Their main difference is that the additive method is preferred

when the seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportionally to the level of the series. This paper applies Holt-Winters multiplicative method to forecast the local demand for all products in Section 3.2.1 because their time series trend patterns are not constant.

2.3.4 Seasonal Autoregressive Integrated Moving Average Models(SARIMA)

Seasonal Autoregressive Integrated Moving Average Models(SARIMA) explicitly supports univariate time series data with a seasonal component.[3]

Def 2.6 (ARIMA)**.** ARIMA is an acronym for Auto Regressive Integrated Moving Average. The full model can be written as

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

where y'_t is the differenced series (it may have been differenced more than once).

The "predictors" on the right hand side include both lagged values of y_t and lagged errors. It is called an ARIMA(p, d, q) model, where p is the order of the autoregressive part; d is the degree of first differencing involved; q is the order of the moving average part.

Def 2.7 (Seasonal ARIMA)**.** A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models we have seen so far. It is written as follows:

$$\text{ARIMA}\ \underbrace{(p, d, q)}_{\substack{\text{Non-seasonal} \\ \text{part of the model}}} \qquad \underbrace{(P, D, Q)_m}_{\substack{\text{Seasonal part} \\ \text{of the model}}}$$

where m = number of observations per year.

Wang et al.[40] adopted SARIMA to predict the monthly precipitation for the coming three years and the model fitted well. They pointed out that the SARIMA model was a proper method for modeling and predicting the time series of monthly data. As for product demand, Srikanth Sankaran[35] forecasts the daily demand for the fresh vegetable product (onions) in a Mumbai wholesale market, based on historical data using SARIMA, which outperformed other contenders in terms of forecasting accuracy on both in-sample and two out-of-sample datasets of the models developed and tested. This paper adopts SARIMA to predict the monthly overseas demand for all products because of significant seasonality in Section 3.2.2.

2.3.5 Knapsack problem

The knapsack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.[29] An example is shown in Fig. 6.

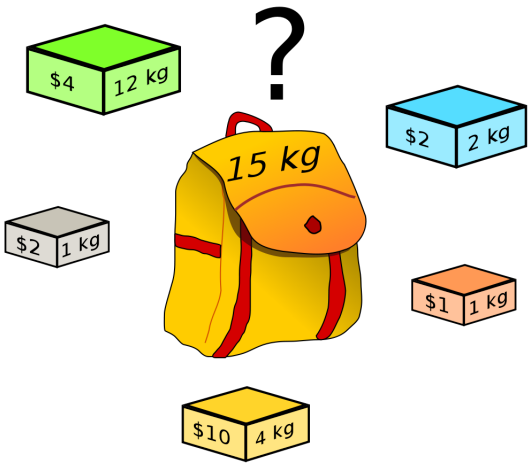


Figure 6: An example of knapsack problem[15]

The problem in Section 3.3 is formulated similarly to a knapsack problem.

2.3.6 Greedy algorithm

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It does not worry whether the current best result will bring the overall optimal result. The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.[25] The basic idea of the greedy approach is to calculate the ratio value/weight for each item and sort the item on basis of this ratio. Then take the item with the highest ratio and add them until the next item cannot be added as a whole. An illustration of how to use greedy algorithm to solve the knapsack problem in Fig.6 is displayed in Fig.7.

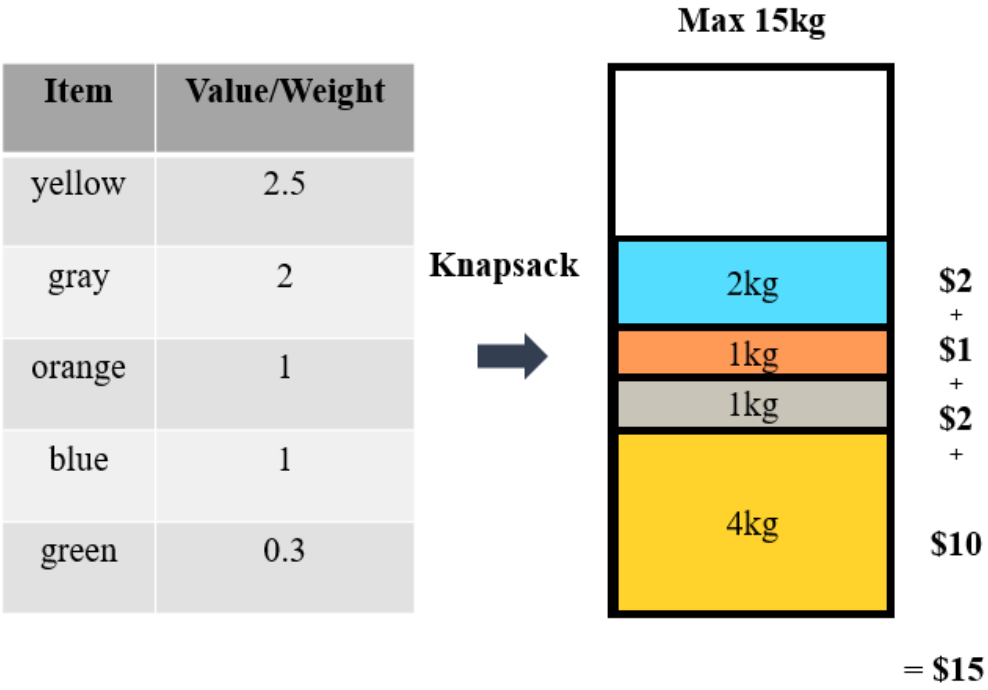


Figure 7: An example of solving knapsack problems by greedy algorithm

A greedy algorithm in Section 3.3.2 is applied to speed up a mixed-integer linear programming model shown in Section 3.3.1.

2.3.7 Bin packing problem

The bin packing problem is a well-studied problem in combinatorial optimization. In the classical bin packing problem, there is a list of real numbers in $(0, 1]$ and the goal is to place them in a minimum number of bins so that no bin holds numbers summing to more than 1. [14]

Def 2.8. Given a list $A = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ of real numbers in the range $(0, 1]$, find the minimum number N of "bins" for which there is a mapping $f : [1, ..., n] \to [1, ..., N]$ such that for all i, the sum of those α_j for which $f(\alpha_j) = i$ does not exceed 1. This least N is termed $A^\star$.

An example is shown in Fig. 8, where there are many cubes of volume less than one on the left. The question is what the minimum number of bins with the size of one needed to pack all these cubes is.

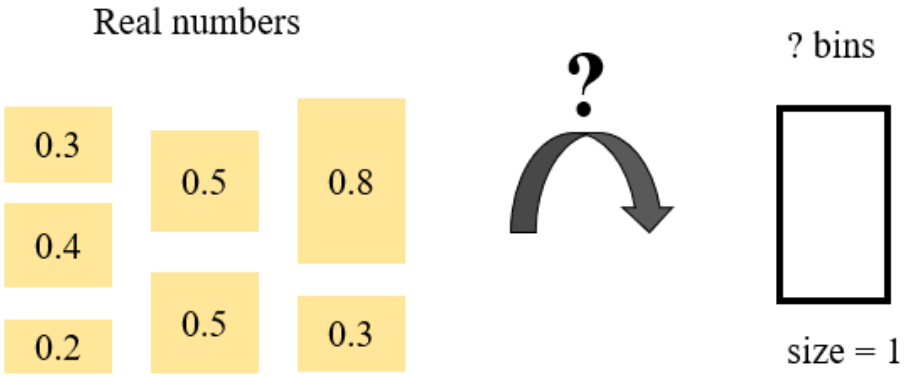


Figure 8: An example of bin packing problem

The bin packing problem is well-known to be NP-hard. [20]In 1972, Garey, Graham, and Ullman [19], provided a detailed analysis of four heuristics: First Fit, Best Fit, First Fit Decreasing (FFD), and Best Fit Decreasing (BFD) to give an approximate solution of this problem.

(i). First Fit. Initially, all bins are "filled to level" 0. Consider $\alpha_1, \alpha_2, \cdots, \alpha_n$ in that order. To consider α_i find the least j such that B_j is filled to a level $\beta \leq 1 - \alpha_i$. Place α_i in B_j. B_j is now filled to level $\beta + \alpha_i$.

(ii). Best Fit. Initially, all bins are filled to level 0. Consider $\alpha_1, \alpha_2, \cdots, \alpha_n$ in that order. To consider α_i find that bin B_j such that B_j is filled to level $\beta \leq 1 - \alpha_i$ and $\beta + \alpha_i$ is as large as possible. Place α_i in B_j. B_j is now filled to level $\beta + \alpha_i$.

(iii). First Fit Decreasing (FFD). Order $(\alpha_1, \alpha_2, \cdots, \alpha_n)$ largest first, and then apply (i).

(iv). Best Fit Decreasing (BFD). Order $(\alpha_1, \alpha_2, \cdots, \alpha_n)$ largest first, and then apply (ii).

In this paper, a bin packing problem is formulated in Section 3.4.1, and Best Fit Decreasing (BFD) is adopted to solve the problem in Section 3.4.2 because it performs the best among the four heuristics.

3 Methodology

The problem will be addressed in the following steps in order to answer the research questions and achieve maximal profit. For the first research question, an initial step is to exclude those products that cannot make a profit because they do not bring any benefit to the income. Next, the monthly demand for each product of the next year is predicted based on historical data in order to answer the second question. In terms of the third question, two mixed-integer linear programming (MIP) models are built, one is to find a placement schedule of the products in the farm of a fixed number of racks, and the other is to minimize the number of robots and find out how robots can be arranged to serve all racks in a day. Eventually, the optimal farm size and the number of racks are determined by gradually increasing the farm size and repeatedly running the two MIP models, after which the corresponding profit of each farm is calculated. The optimal farm size should balance the product income, the farm cost, and the robot cost.

3.1 Selection of products

Before the demand prediction step, it is necessary to select products worth being planted on a vertical farm, and therefore the products that do not make a profit should be excluded in order to prevent losses. The retail price of every product is known but the cost of planting them is dependent on the number of robots serving them, which requires information on how they are arranged on racks. Although how these products will be arranged on racks remains unclear, it is possible to get an estimated cost of a given product per rack by filling the rack with the same products on all shelves. Given the following notations.

- $G = \frac{150}{(30*24*60)}$: robot cost per minute
- c_p: the growth cycle length for product p
- f_p: the water frequency of product p
- T_p^w: the watering length of product p each time
- T_p^l: the lighting length of product p each time
- b_p: the retail price of product p (per shelf)
- A_p: the cost of product p per rack
- B_p: the profit of product p per rack

Then the cost of a given product per rack and per shelf can be estimated as follows.

$$A_p = \text{water cost} + \text{light cost} = G \times c_p \times [f_p \times (90 + 10 \times T_p^w) + (90 + 10 \times T_p^l)]$$

$$B_p = 10 \times b_p - A_p$$

Then only those products with a profit of more than 0, i.e., $B_p > 0$ are selected. This reduces the number of products from 105 to 28. Among all these products, the maximum cycle length is 25 days (less than a month), which means it is possible to grow a tray of any product within a month.

3.2 Demand prediction

In order to better organize farming for the coming year, it is necessary to use appropriate methods to forecast local and overseas demand for various products based on historical data. The distribution of local demand for all products behaves in a similar way and so it is for the overseas demand. However, the time series of local demand for all products fluctuate very differently from overseas, where the former shows no obvious patterns and looks like a white noise process, while the latter presents a very evident seasonal pattern, and therefore these two demands need to be predicted separately by different methods.

3.2.1 Local demand

Fig. 9 shows that local demand for cabbages appears stationary, fluctuating within a range of two hundred above and below 640. It is also the case for the local demand for other products.

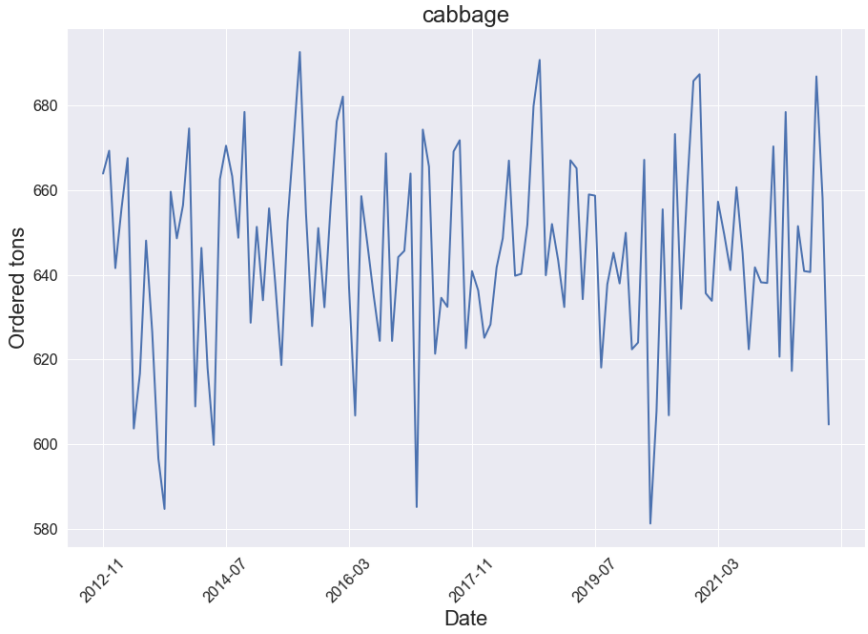


Figure 9: Time series of local demand for cabbages from 2012 to 2021

The time series of every product is tested to be stationary by Augmented Dickey-Fuller test[13] with a p-value less than 0.05. In addition, the times series of all products, except 7 shown in Fig. 10 are tested by Ljung–Box test[41] to be a white noise process for lags from 1 to 12.

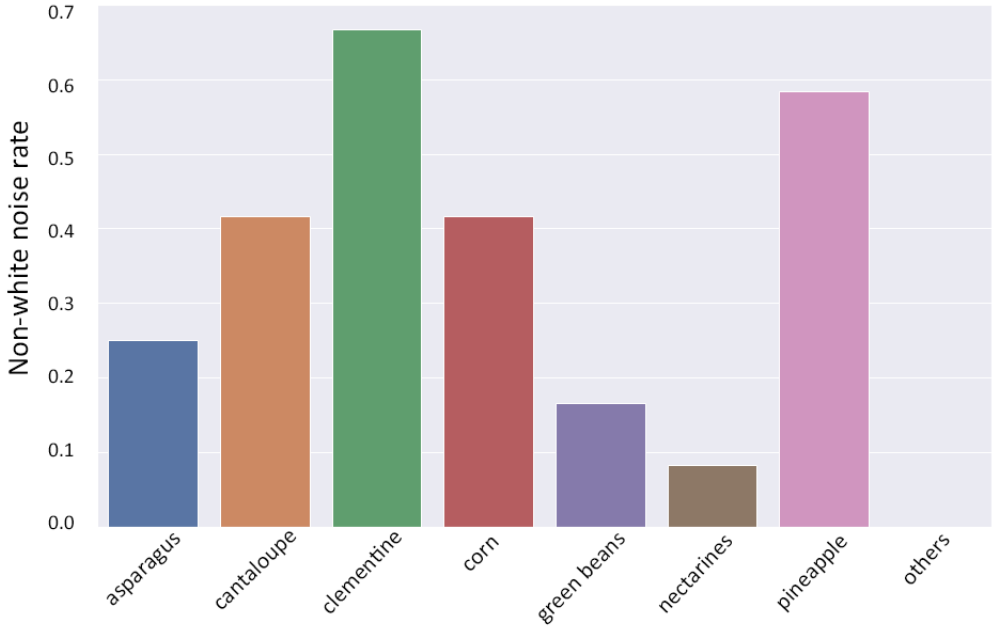


Figure 10: Non-white noise rate of all products from lag 1 to 12

The time series of every product has been tested whether they are white noises from lag = 1 to 12. The non-white noise rate is defined as the ratio of the number of lags one time series is tested to not

be white noise process and 12. For instance, the time series of *asparagus* are tested not to be a white noise process when lag $= 3, 6, 8$, and therefore its non-white noise rate is $3/12 = 0.25$. The lower the non-white noise rate of a product is, the higher possibility its time series is a white noise process.

Fig. 10 displays non-white noise rate for all products. The non-white noise rate of all products other than *asparagus*, *cantaloupe*, *clementine*, *corn*, *green beans*, *nectarines*, *pineapple* is 0, so their time series are deemed as white noise processes. The non-white noise rate of nectarines is less than 0.1, so its time series is considered to be a white noise process as well. As for the mentioned 6 products, there is a possibility that their time series are white noise processes, but to confirm this, it is essential to test whether the time series has auto-correlation for more lags.

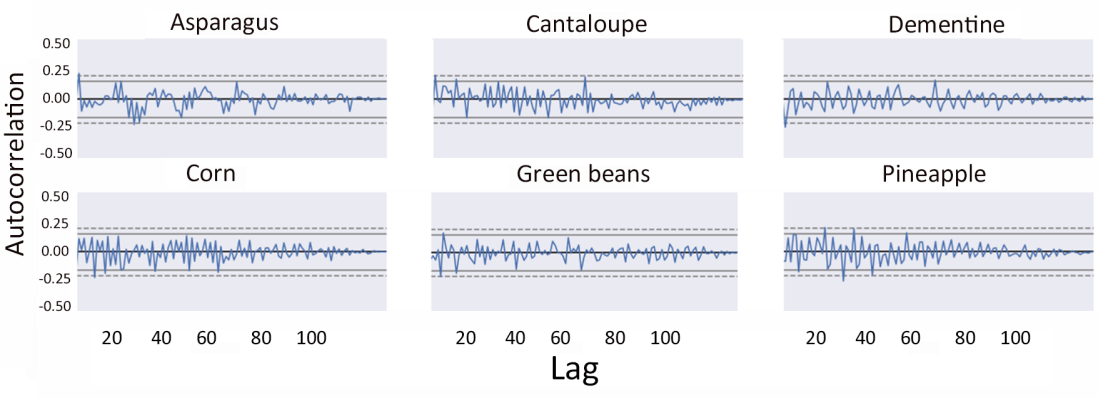


Figure 11: Autocorrelation of local demands for 6 products

Fig. 11 shows autocorrelation of them with lags increasing from 1 to over 100. The autocorrelation plots do not show any significant auto-correlation features. Confidence levels at the peaks were at 75%, but this was only a statistical fluke. Consequently, it can be assumed that the demands of all products are white noise processes.

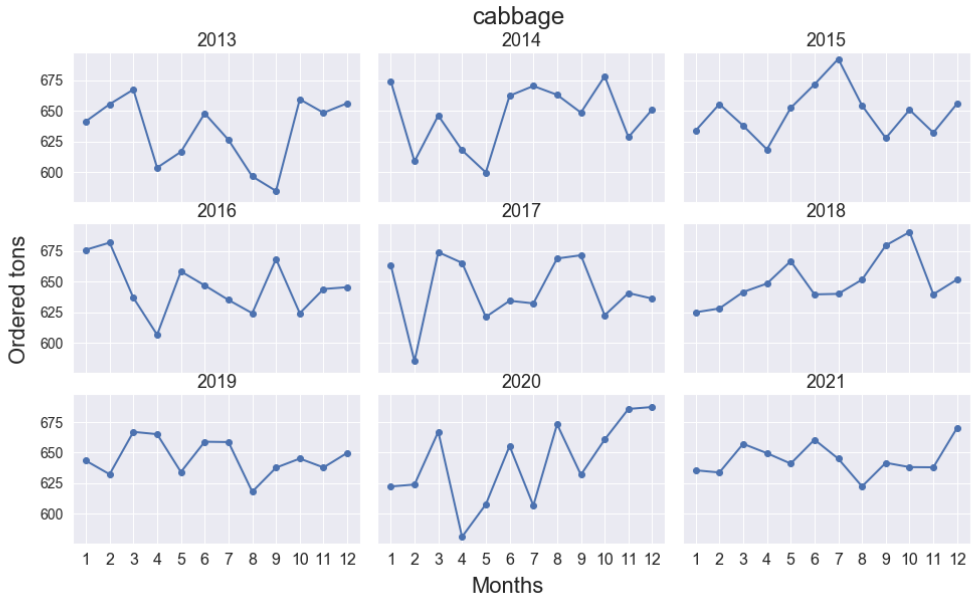


Figure 12: Time series of local demand for cabbages

The predicted monthly demand for each product in the coming year can be assumed as the average value of the past. However, taking the average value as a prediction is not an ideal choice because it can lead to biased estimators. Fig. 12 display the local demand for cabbages each year. Although there is no obvious seasonal pattern, after removing the trend, the seasonal pattern is detected in Fig.

13 by seasonal decomposition.

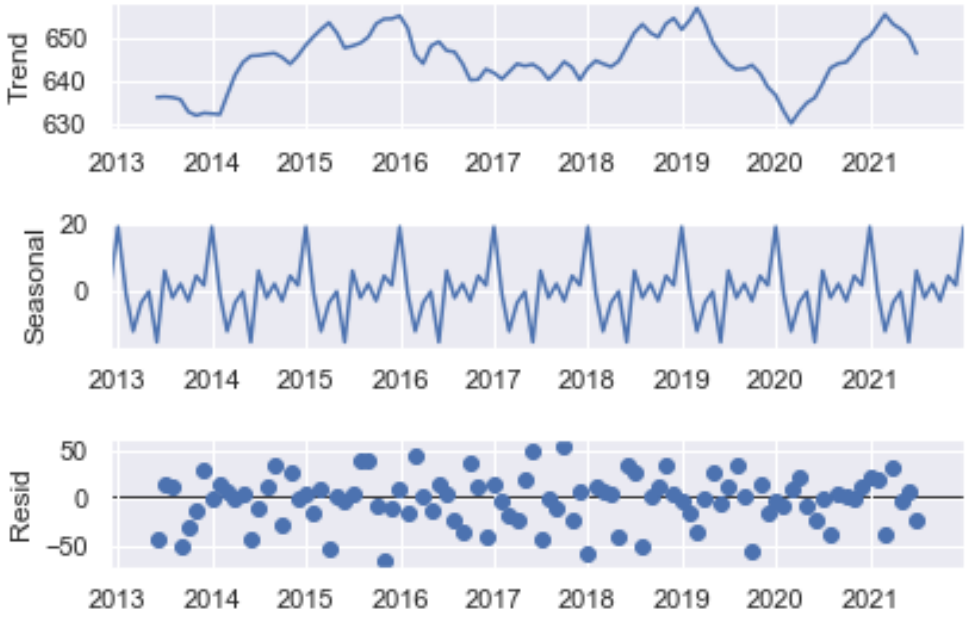


Figure 13: Seasonal decomposition of local demand for cabbages

Under this circumstance, Holt-Winters multiplicative method is applied to forecast the demand for each product because it has both a proportional trend and a seasonality. As for the cross-validation method, since k-fold does not consider the factor of time order, there are two *Nested Cross-Validation*[39] methods to measure the results of time series prediction, namely *Predict Second Half* and *Day Forward-Chaining*, as shown in Fig. 14 and 15.

Predict Second Half is adopted due to the fact that the ratio of predicted data and observed ones, $\frac{1}{9}$ is almost equivalent to that of the training set and test set, $\frac{1}{8}$.

Figure 14: Predict Second Half[8]

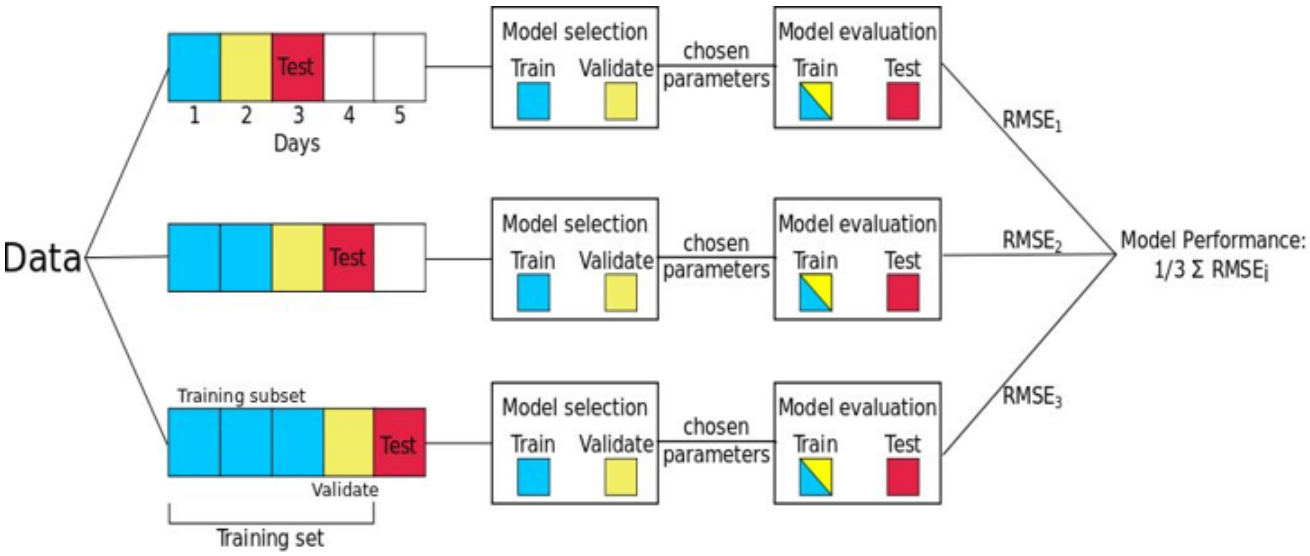


Figure 15: Day Forward-Chaining[8]

In comparison, the unbalanced ratio of the training set and test set may affect the results. There are many metrics that can be used to evaluate the error, such as Mean Square Error (MSE), Root Mean Square Error(RMSE), Mean Absolute Error(MAE), and Mean Absolute Percentage Error (MAPE). MAPE is a relative error percentage with a fixed range of $[0, 1]$ while the range of other metrics is $[0, +\infty]$. Since the quantity of local demand is different from that of overseas, adopting MAPE can provide a relatively direct comparison between the two regardless of the quantity. The average MAPE of all products is 3.84%. An example of cabbages is shown in Fig. 16 with MAPE of 3.84%.

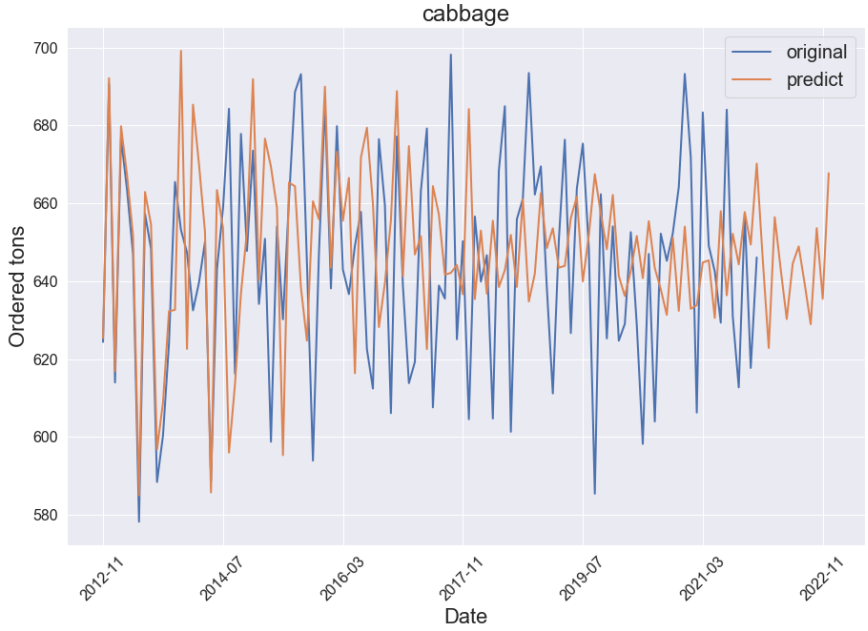


Figure 16: Forecast time series of local demand for cabbages

3.2.2 Overseas demand

The overseas demand for lettuce in the past is shown in Fig. 17 and Fig. 18.

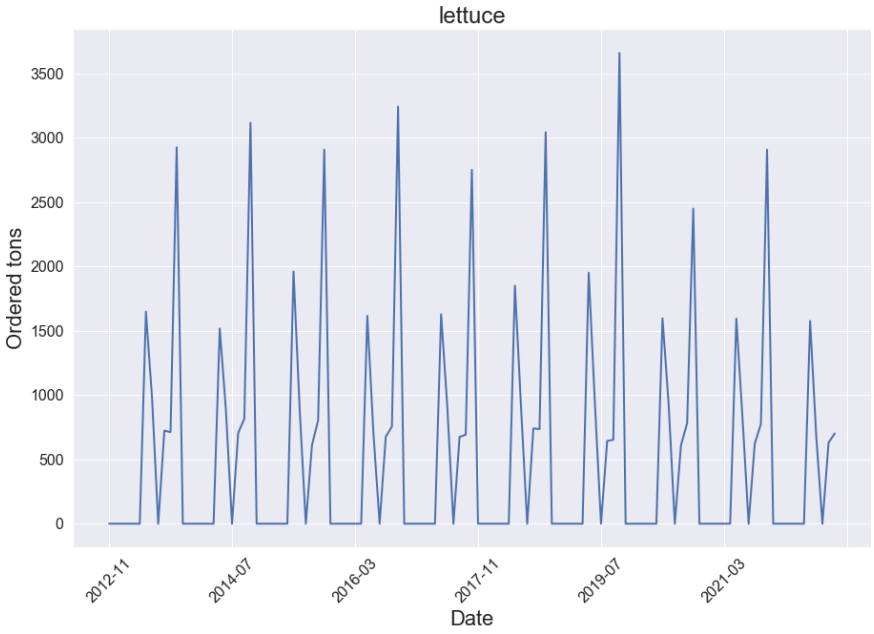


Figure 17: Time series of overseas demand for lettuce

Fig. 17 shows that overseas demand for lettuce shows a very significant seasonality. The same conclusion can be drawn from Fig. 18 since it can be seen that the patterns of all time series are very similar. This is also the case for the overseas demand for all other products.

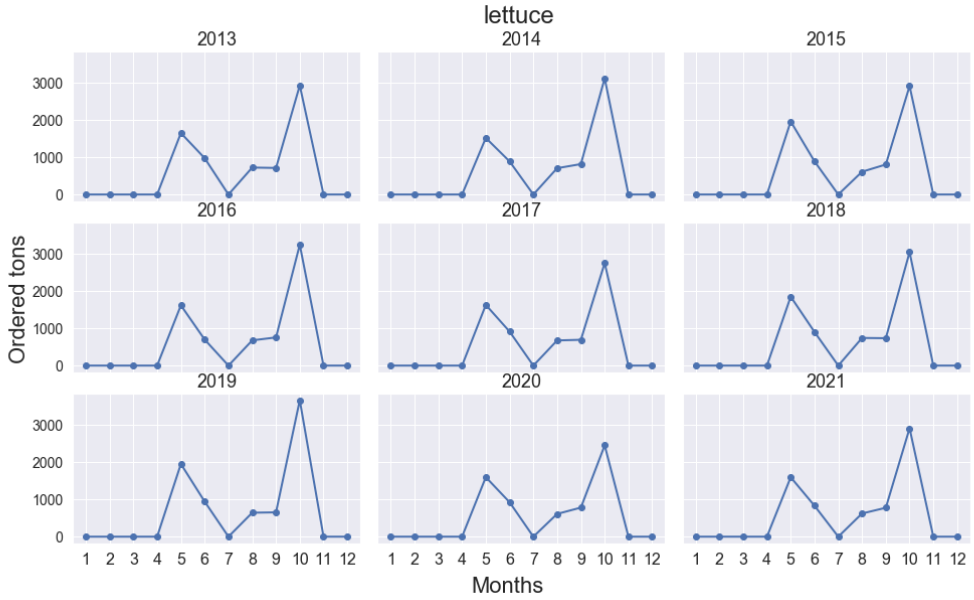


Figure 18: Separate time series of overseas demand for lettuce from 2012 to 2021

Due to eminent seasonality, SARIMA[3] is adopted to predict the monthly overseas demand for all products. After removing seasonality from the time series (s = 12) and making one difference (d = 1), the shifted time series, autocorrelation function (ACF) and partial autocorrelation function (PACF) are plotted in Fig. 19. It can be told from Fig. 19 that the shifted time series is stationary with a p-value less than 0.05. By observing the ACF and PACF and comparing the magnitude of the AIC for all possible parameters, the following parameters were finally determined as follows.

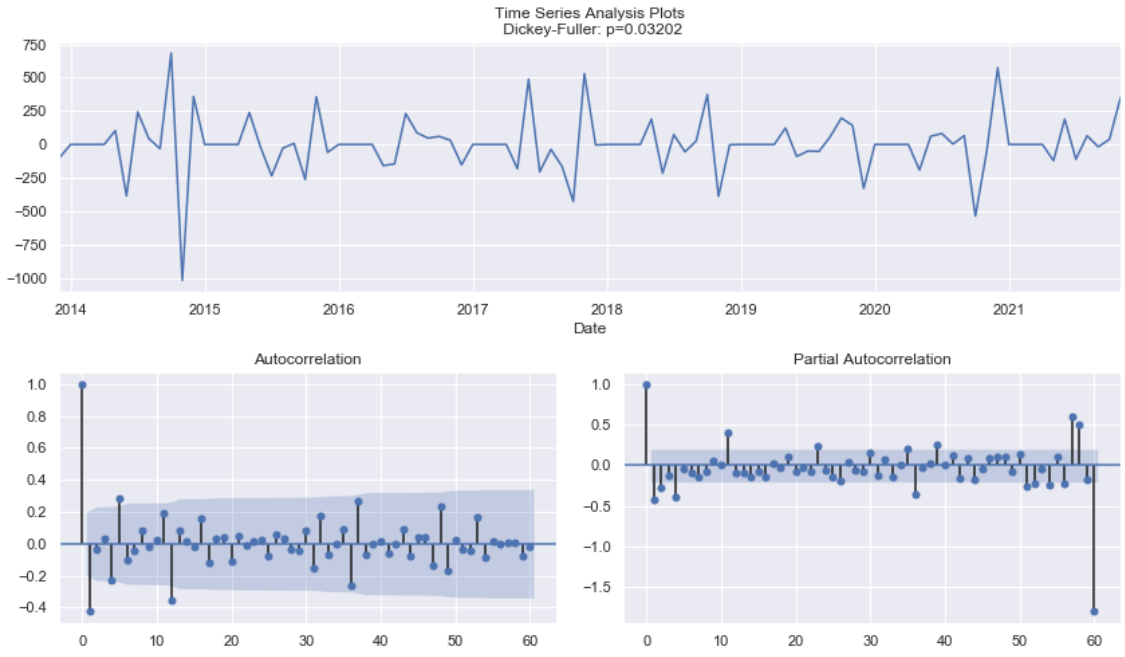


Figure 19: ACF and PACF of overseas demand for lettuce

As for trend elements, trend autoregression order (p = 0), trend difference order (d = 1), trend moving average order (q = 2). In terms of seasonal elements, seasonal autoregressive order (P = 0), seasonal difference order (D = 1), seasonal moving average order (Q = 1), the number of time steps for a single seasonal period (s = 12).

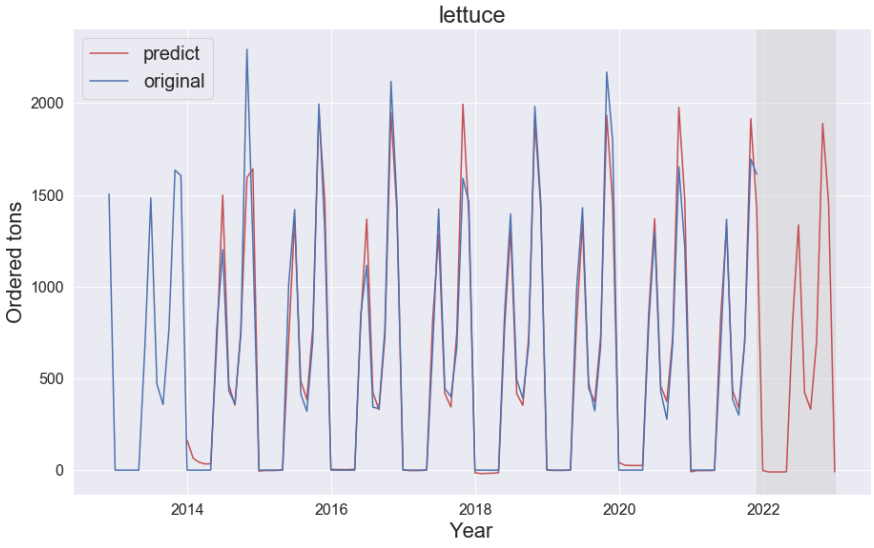


Figure 20: Forecast of overseas demand for lettuce in the next year

With the parameters mentioned above, a SARIMA model is built to predict the overseas demand for lettuce in the next year, as shown in Fig. 20. The overseas demand of all products are forecast by SARIMA as well, with each own specific parameters. The average MAPE of the test set of *Predict Second Half* method is 11.79%.

3.3 Optimization of monthly rack arrangement

This section develops a MIP model to solve the problem of finding a placement schedule for the products in the farm of a fixed number of racks each month. For a given month, the known parameter is the number of racks and the demand for all products. The model provides a solution that tells what product should be placed on each shelf every day. It works as follows.

If the number of racks is fixed, the next step is to check whether it is enough for all demands of this month. If it is not, then select a subset of all demands that makes the most profit to be fit into the racks, as described in the second and third steps of Algorithm 1. For a fixed number of racks and product demands, Model A in section 3.3.1 can provide a solution to satisfy the requirement of putting products that have less frequency of water on the top, and vice-versa, as introduced in the problem description section.

One schematic diagram of the problem in February is shown in Fig. 21, where the number of racks is $\frac{N}{10}$ (fixed) with N shelves. There are two terms in the figure introduced as follows.
Terminology

- tray: a kind of products that grows on a shelf for a full cycle.
- slot: an available day of a shelf. For example, a tray of lettuce (growth cycle = 7 days) takes 7 slots, i.e., one shelf for 7 days.

The first row of the table in Fig. 21 can be interpreted as the total demand of the first product oranges is 37 trays. The cycle length of orange is 1 day and the water frequency is 8 times per day. On the right of the figure there are N shelves in $\frac{N}{10}$ racks of which a rack contains 10 shelves from level 1 to 10. Since it is in February, there are 28 days in the month so the number available slots of each shelf is 28. The question is how to arrange each tray of product in those shelves of different levels.

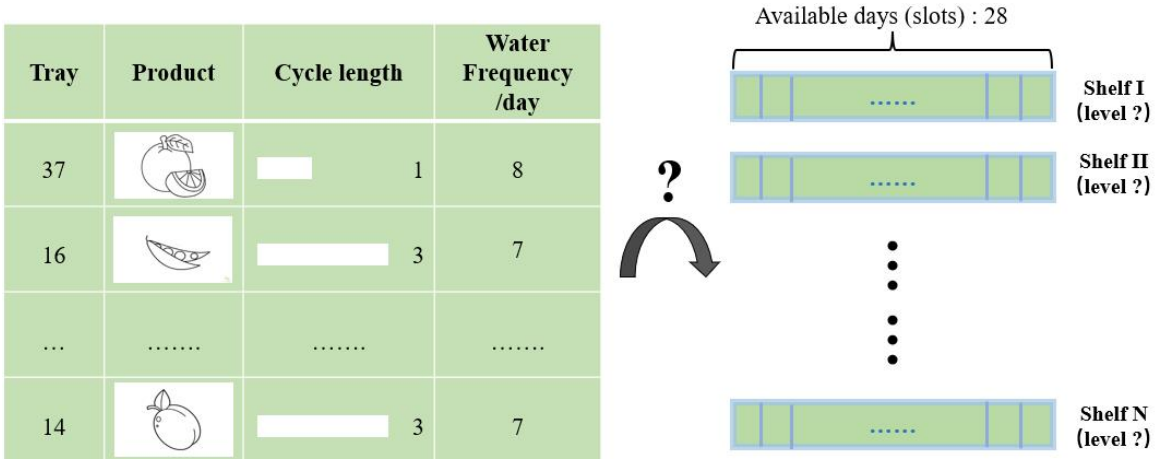


Figure 21: The schematic diagram of rack arrangement in February

Since the number of racks is fixed, this problem can be seen similarly as a knapsack problem [29]. In fact, if the water frequency of each product is envisioned as its value, and cycle length as its size. Likewise, a rack is taken as a knapsack, and different shelf levels as different pocket levels. An additional rule is that when a product is placed in a pocket, its value will be its value multiplied by the coefficient corresponding to the pocket level. The lower the level of a pocket, the higher the corresponding coefficient. Then the question becomes an adjusted knapsack problem, which is how should all trays of products be placed into these knapsacks in order to maximize the total value, given a fixed number of knapsacks and a fixed set of products.

3.3.1 Model formulation (Model A)

Given the following additional notation for parameters,

- D: the number of days in the given month

- M: the given number of racks
- P: the set of selected products
- $p \in P$: product p
- $R = \{1, 2, ..., M\}$: the set of all racks
- $r \in R$: rack r
- $L = \{1, \cdots, 10\}$: set of all shelf levels
- $l \in L$: shelf level l
- I_p: the set of total trays in product p
- $i_p \in I_p$: the i^{th} tray of product p
- c_p: the growth cycle length for product p
- f_p: the water frequency of product p

Set the variable as

$$x_{i_p,r,l} = \begin{cases} 1, & \text{if the } i^{th} \text{ tray of product } p \text{ is allocated to the } l^{th} \text{ level shelf of rack r} \\ 0, & \text{otherwise} \end{cases}$$

for each $i \in I_p$, $p \in P$, $l \in L$ and $r \in R$.

The model of monthly rack arrangement (Model A) is formulated as follows,

$$\max \sum_{r \in R} \alpha_l \sum_{l \in L} \sum_{p \in P} \sum_{i_p \in I_p} f_p \cdot x_{i_p,r,l} \tag{1}$$

$$\text{s.t.} \quad \sum_{p \in P} c_p \Big(\sum_{i_p \in I_p} x_{i_p,r,l} \Big) \leq D \qquad \forall r \in R, l \in L \tag{2}$$

$$\sum_{r \in R} \sum_{l \in L} x_{i_p,r,l} = 1 \qquad \forall p \in P, i_p \in I_p \tag{3}$$

where $\alpha = (10, 9, \cdots, 1)$ is the coefficient to provide benefit for each product placed in different shelves.

Objective (1) ensures that the products with high water frequency are placed at the bottom and vice versa because it maximizes the total summation of weighted frequencies where the higher shelf level a tray of product is allocated to, the less weight its frequency multiplies. Constraint (2) limits the summation of the growth cycle lengths of all products placed on each shelf to the number of days in the month. Constraint (3) ensures that each tray of products can only be allocated to at most one shelf.

This model calculates which products should be planted on which shelf of each rack but does not tell the chronological order of planting products on the same shelf in this month, i.e., which tray should be planted before others. To ensure the rule that the water frequencies of products placed on each rack are increasing from top to bottom every day, the products with the largest frequencies will be planted first and then the second largest. For instance, if two trays of basil (frequency = 6, growth cycle length = 13) and one tray of mushrooms (frequency = 8, growth cycle length = 4) are calculated by Model A to be placed on the second shelf of rack 1 in January (days = 31), then their order should be planting a tray of mushrooms for the first four days, and then planting two trays of basil for the next 26 days. For the following days, the shelf remains empty.

An example result of one rack is shown in Fig. 22, where each colour represents a product, and the numbers $f(c)$ in each bar mean frequency and cycle length respectively. A red bar with 11(4) represents a tray of strawberries with a growth cycle of 4 days and a water frequency of 11 times per day. The row displays the shelf level of this rack and the column represents each day (slot) of the rack.

The red bar on the bottom left can be understood as a tray of strawberries should be planted on Shelf 1 of this rack for the first four days. This example displays how products with different frequencies are placed on different shelf levels of one single rack. The chronological order of planting products on the same shelf this month is manually arranged from the largest frequency to the smallest as described above. The most important thing is that, for every single day, products with high frequencies are placed at the bottom shelf and those with low frequencies are placed at the top. The frequency of products from top shelf to the bottom of this rack is in increasing order every day.

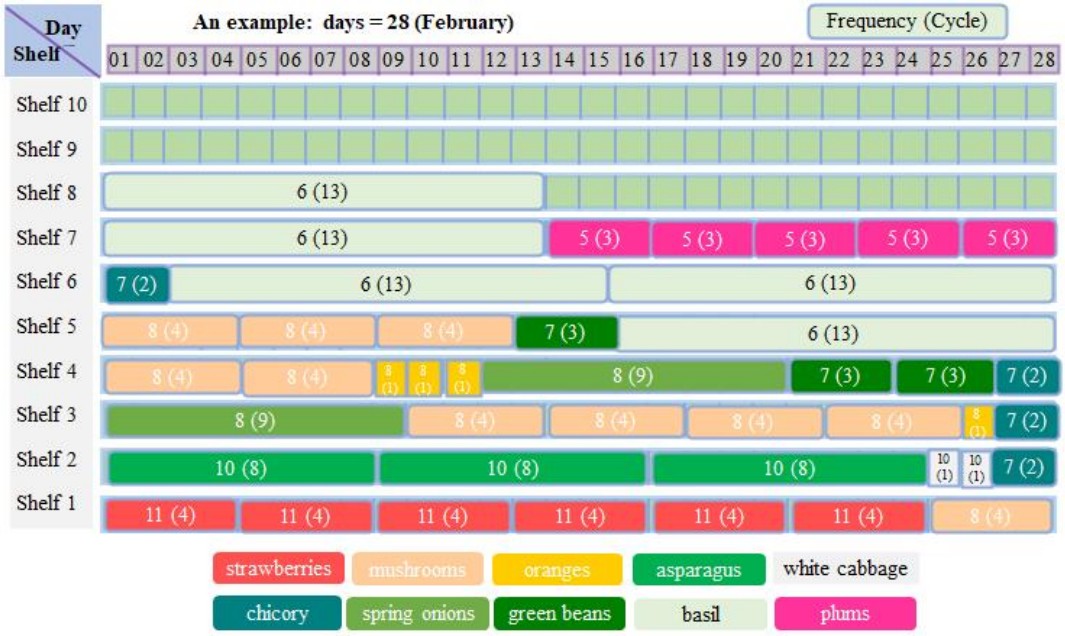


Figure 22: An example of the arrangement for one rack in February

3.3.2 Greedy algorithm: A speed-up solution for Model A

When the number of racks is large and the demand for product set is high, the linear programming software, such as *Cplex*, *Gurobi* and *Xpress* can be very slow in providing results to model A. A greedy algorithm, Algorithm 1 is proposed to speed up the whole process by providing a highly approximate solution. This algorithm simulates the process of manually placing each product into all racks one by one according to the water frequency. The idea of this greedy algorithm is to first order all trays of products according to its water frequency from the largest to the smallest, and then place every tray of product on the lowest empty shelf which it can fit into.

Example results of the first-day placement of 25 racks in January obtained by Model A and greedy algorithm are displayed in Table 4 and Table 5 in Appendix respectively. The water frequencies of each rack are in increasing order from bottom to top in both tables, but the greedy algorithm takes needs more shelves to place all products because there exist some wasted slots compared to Model A. However, although the greedy algorithm does not provide an optimal placement schedule of Model A, the result is highly approximate, where only 8 extra shelves are planted with products.

Algorithm 1: Greedy algorithm to solve model A

Input: Month m, with D days;
Number of racks M, and rack set $R = \{1, \cdots, M\}$;
Shelf level set of each rack $L = \{1, 2, \cdots, 10\}$;
Product set P with trays $|I_p|$, cycle length c_p, water frequency f_p of each product p;
Estimated profit each product makes per rack O_p;
Output: The placement of each tray (i_p) of each product $p \in P_m$;

Reduce product set P to P_m s.t. for any $p \in P_m$, the demand of p is greater than 0 ($|I_p| > 0$);
Sort all products in P_m in a descending order according to the profit per rack O_p;
Select top a products s.t. $P_m^{'} = \{p_1, \cdots, p_a\}$ can be fit into M racks, and $\{p_1, \cdots, p_{a+1}\}$ cannot. (If the racks are enough then $P_m^{'} = Pm$);
Sort all products in $P_m^{'}$ in a descending order according to frequency f_p and cycle length c_p;
Initialization: Set the number of available slots of each shelf level of each rack as D ($G_{r,l} = D$);
Set the products placed in each shelf level of each rack as empty set ($J_{r,l} = \emptyset$);
for *each* $p \in P_m^{'}$ **do**
    left $= |I_p|$; // keep track of the number of trays not placed yet of product p
    /* start from the first shelf level of all racks */
    for $l \in L$ **do**
        for $r \in R$ **do**
            if $G_{r,l} \geq c_p$ **then**
                k =min(left, $\lfloor G_{r,l}/c_p \rfloor$) ; // the number of trays can be placed in this shelf
                put k products in $J_{r,l}$;
                left = left $-k$;
                $G_{r,l} = G_{r,l} - k \times c_p$;
                if *left* ≤ 0 **then**
                    /* If all trays of product p get a shelf to be placed, then start loops for the next product */
                    Goto 7;
Return J ; // the placement of all trays in racks

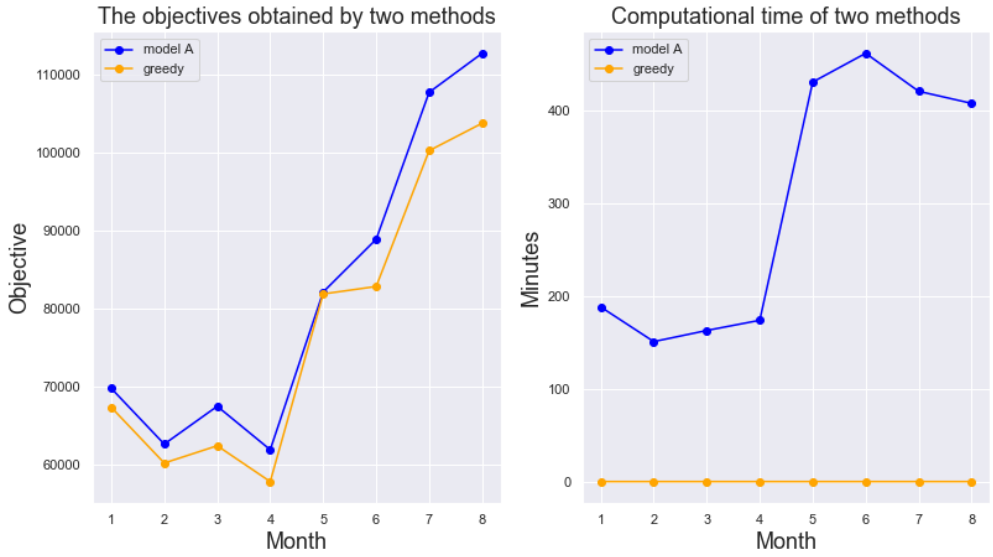


Figure 23: Comparison of the results obtained by Model A and Greedy algorithm (25 racks)

To test the results of the greedy algorithm, the first-day placements of 25 racks obtained by Model A and the greedy algorithm are compared. All testing in this paper is done using the IP solver *Gurobi 9.5.2* with *Pyomo 6.4.1* of *Python 3.7.6*[1] on a computer with 1.8 GHz Intel i7 processor and 16GB of RAM. Due to the limitation of computer configuration, only the first eight months are tested.

The comparison results are displayed in Fig.23. The first subplot in compares the objectives obtained by the two methods and the second subplot compares the computational time of the two. It can be shown that the greedy reduces the objective by at most 8%. However, it takes hours for model A to solve the problem when the number of needed slots increases dramatically but the greedy algorithm only takes a few seconds regardless of the number of needed slots. The monthly needed slots of 25 racks are displayed in Fig.34. Therefore, the greedy algorithm is adopted to speed up the process.

3.4 Optimization of daily robot arrangement

After obtaining the results of how to schedule plants on all racks every day in each month, another mixed integer linear programming (MIP) model is built to solve the daily robot arrangement problem depicted in Fig. 24. There are N (fixed) racks planted with different products on each shelf on the right of this figure, and the problem is how many water robots and light robots are needed respectively and how these robots should be allocated to serve all these racks.

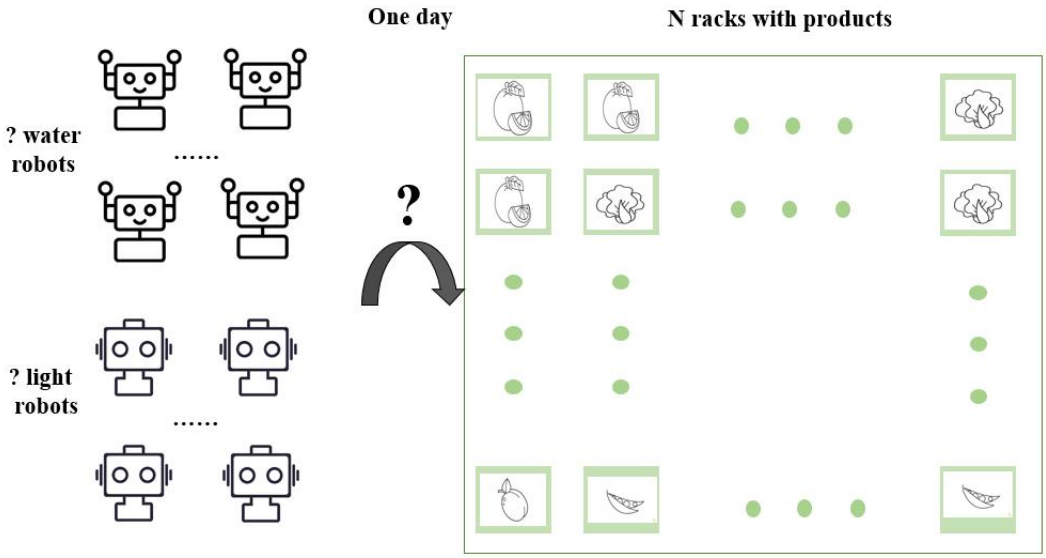


Figure 24: Schematic diagram of robots serving all racks in a day

It is highlighted that the placement of the first day of each month is picked to calculate the minimum number of robots needed in this month because the water frequency of products on each shelf is decreasing as time goes by, and therefore the first day of each month requires the most number of robots. For instance, the first-day placement of the rack in Fig. 22 is shown in Fig. 25, and its total frequency is higher than that of any other day in the month. There is a term that should be introduced to illustrate the problem more clearly.

Terminology

- round: a rack being served by a robot for once.

[1]The source code can be found at `https://github.com/ZhiyingChen/vertical_farm`

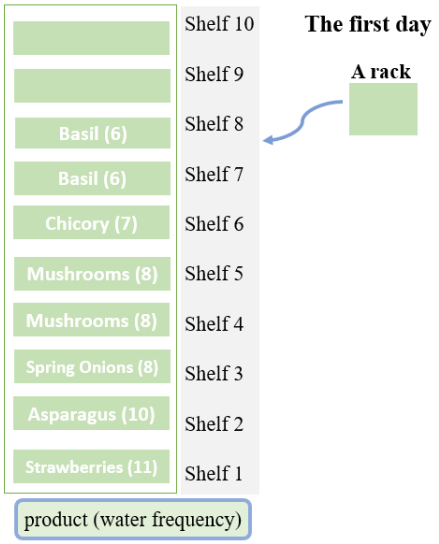


Figure 25: The first-day placement of the rack in Fig. 22

Fig. 26 provides an example of a rack filled with products and the watering time of ten rounds that it needs one day. It is assumed here that each shelf needs watering for one minute per time.

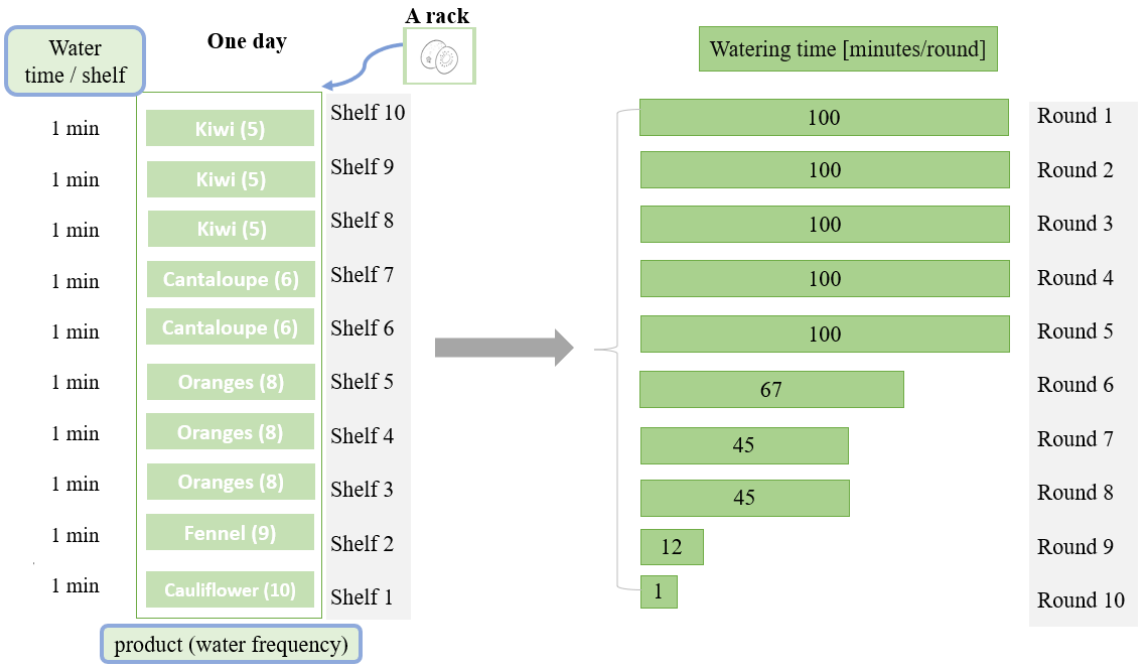


Figure 26: An example of the watering rounds of a rack

Table 3 introduces how to calculate the watering time for each round of the rack in Fig. 26. The product *kiwi* needs watering five times a day, and therefore the rack requires a robot to climb up to the tenth shelf five times. The total time it takes to climb up to the tenth shelf and finish the task of watering all shelves is 100 minutes, including the time of climbing up, watering, and going back down. After doing it for five times, there is no need to water Shelf 8, 9 and 10 any more because they are finished. Then the next round only requires a robot to go to Shelf 7 because there are two times of watering unfinished for the tray of *cantaloupe* on it.

Round	Climb to which shelf	Minutes/Round (Climbing + Watering)
5	10	10*(10-1)+10 = 100
1	7	10*(7-1)+7 = 67
2	5	10*(5-1)+5 = 45
1	2	10*(2-1)+2 = 12
1	1	10*(1-1)+1=1

Table 3: An example of watering time of a rack

It is also the case for the light robot, however, each rack only requires one round to light all shelves per day. Then the problem can be deemed as finding a way to put all rounds, including watering and lighting, of each rack into a working slot of robot so that the total number of robots is minimized. The schematic diagram of the problem is shown in Fig. 27.

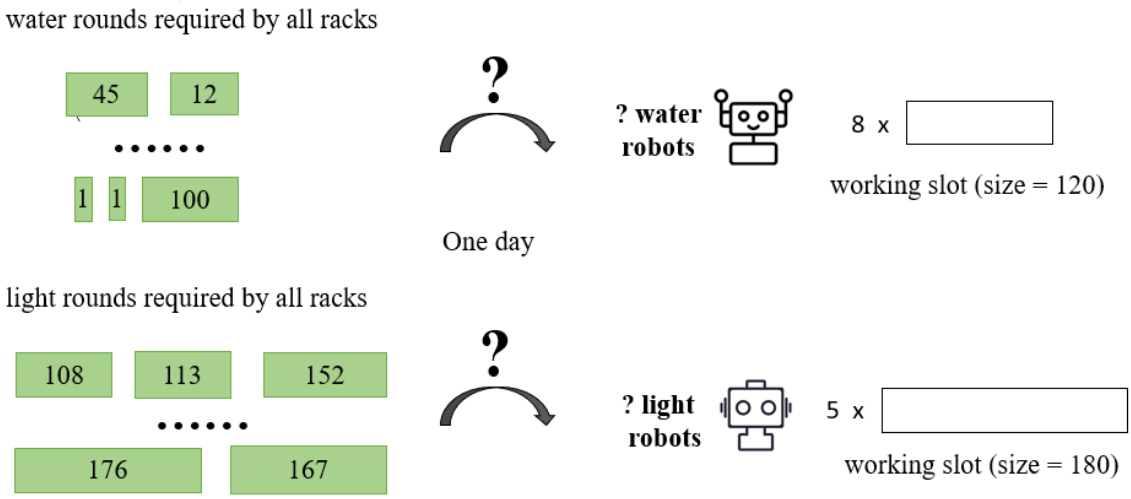


Figure 27: Schematic diagram of robot optimization problem

It is worthwhile to note that the following two assumptions are made.

1. The horizontal moving time of each robot is eliminated.
2. The constraint that a rack cannot be served by multiple robots is loosed, i.e., a rack is allowed to be served by multiple robots.

Assumption 1 is made due to the following two reasons. One is that the time spent to move up/down on a single shelf is much greater than the dislocation on the ground from one shelf to another, and therefore the horizontal moving time is not comparable with the vertical moving time.

The following argument provides the second reason for the elimination of the horizontal moving time. Table 4 and Table 5, where example results of the first-day placement of 25 racks in January obtained by Model A and greedy algorithm are displayed. The rows of these tables indicate each rack and the columns represent each shelf of the corresponding rack. The first row and the first column of Fig. 4 with "strawberries (11)" means a tray of strawberries with a water frequency of 11 is planted on the first day of this shelf.

From these figures, it can be seen that there is a very high rate of the water frequencies of different racks being the same on the first day. If this is the situation for all racks, then it means that for a robot, there is no difference between serving a rack nearby and another rack far away if their placements are the same, and therefore robots only need to move quickly to another rack nearby after finishing one. The following definition and plot are to prove that it is true for all racks.

Define the placement differences between racks as the maximal euclidean distance of the frequency vectors on the first day of the month. Fig. 28 shows the placement differences of the results obtained

from Model A. The maximum euclidean distance is at most 3.8 among all different farms during all months. It indicates that racks have very similar placement regarding frequencies every month.

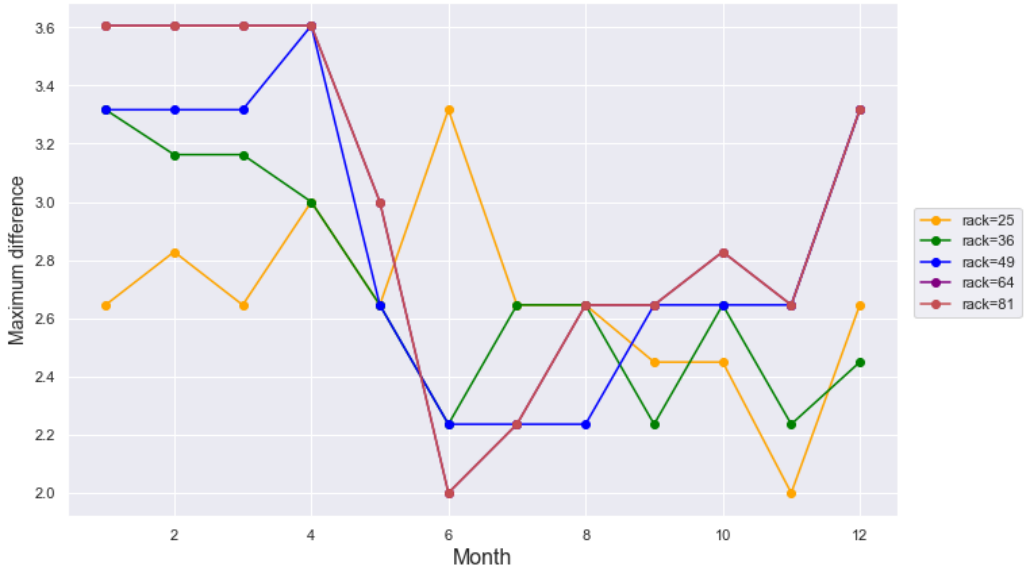


Figure 28: The monthly maximal differences of different farms

As for assumption 2, the example in Table 3 indicates that the total time of serving a rack takes almost the full working hours of a robot so there are few slacks, and therefore it gets almost impossible to spread so many rounds of watering time and lighting time in different working slots. Considering this constraint will cause the model into trouble, which is why it is eliminated.

Then, this problem becomes a typical bin packing problem.[14] The model can be built as in the following section. It is worthwhile mentioning that, before running the model, a number of potentially needed robots should be calculated in advance, and then the scheduling optimization model will reduce it to an optimal number.

3.4.1 Model formulation (Model B)

Given the following additional notations for parameters,

- R: the set of all racks used
- $r \in R$: rack r
- W: the set of potentially needed robots for watering
- $w \in W$: water robot w
- U: the set of potentially needed robots for lighting
- $u \in U$: light robot u
- $S_w = \{1, \cdots, 8\}$: the set of daily working slots of water robot w
- $s \in S_w$: the working slot s of water robot w
- $S_u = \{1, \cdots, 5\}$: the set of daily working slots of light robot u
- $s \in S_u$: the working slot s of light robot u
- $F = \max_{p \in P} f_p$
- $V = \{1, \cdots, F\}$: the set of rounds for each rack

- $v \in V$: the v^{th} round
- h_w: the working length of each working slot of water robot
- h_u: the working length of each working slot of light robot
- $t^w_{r,v}$: the watering time a water robot needs to spend on the v^{th} round of rack r
- t^u_r: the lighting time a light robot needs to spend on rack r

Set the variables as

$$\overline{x}_{r,v,w,s} = \begin{cases} 1, & \text{if the } v^{th} \text{ round of rack } r \text{ is allocated to the } s^{th} \text{ working slot of water robot } w \\ 0, & \text{otherwise} \end{cases}$$

for each $r \in R$, $v \in V$, $w \in W$, $s \in S_w$.

$$\underline{x}_{r,u,s} = \begin{cases} 1, & \text{if rack } r \text{ is allocated to the } s^{th} \text{ working slot of light robot } u \\ 0, & \text{otherwise} \end{cases}$$

for each $r \in R$, $u \in U$, $s \in S_u$.

$$\overline{y}_w = \begin{cases} 1, & \text{if water robot } w \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

for each $w \in W$.

$$\underline{y}_u = \begin{cases} 1, & \text{if light robot } u \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

for each $u \in U$.

The model of daily robot arrangement (Model B) is as follows,

$$\min \sum_{w \in W} \overline{y}_w + \sum_{u \in U} \underline{y}_u \tag{4}$$

$$\text{s.t.} \quad \sum_{r \in R} \sum_{v \in V} t^w_{r,v} \cdot \overline{x}_{r,v,w,s} \le h_w \cdot \overline{y}_w \quad \forall w \in W, s \in S_w \tag{5}$$

$$\sum_{r \in R} t^u_r \cdot \underline{x}_{r,u,s} \le h_u \cdot \underline{y}_u \quad \forall u \in U, s \in S_u \tag{6}$$

$$\sum_{w \in W} \sum_{s \in S_w} \overline{x}_{r,v,w,s} = 1 \quad \forall r \in R, v \in V \tag{7}$$

$$\sum_{u \in U} \sum_{s \in S_u} \underline{x}_{r,u,s} = 1 \quad \forall r \in R \tag{8}$$

Objective (4) is to minimize the number of robots used. Constraint (5) and (6) stipulate that, for any working slot of any robot, the time of all rounds allocated to it cannot exceed its time limit. For instance, the time limit of a working slot of a water robot is 120 minutes, then the time of all rounds allocated to each working slot of a water robot cannot exceed 120 minutes. Constraint (7) and (8) regulate that every round of each rack must be placed at exactly one working slot.

3.4.2 Best Fit Decreasing (BFD) algorithm: A heuristic to solve Model B

For a given rack, the number of rounds it needs to be served by robots per day is determined by the maximal frequency of this rack, so the speed of running Model B by LP solvers becomes extremely

slow as the number of racks increases. Similarly, a heuristic called Best Fit Decreasing (BFD) [19] is generated to speed up the running process of Model B.

The idea of Best-fit decreasing (BFD) is to first sort the numbers in decreasing order and then assign each number in turn to the fullest bin in which it fits. The algorithm runs in $O(nlogn)$ time.[27]

Algorithm 2: Best Fit Decreasing (BFD) algorithm to solve model B

Input: List of lengths of all rounds $T = [118, 98, \cdots, 108]$;
Length of each working slot of a robot h;
Number of working slots of each robot per day K;
Output: Minimum number of needed robots N;
The allocation of every working slot of all robots S;

Sort the rounds of T in decreasing order;
Initialization: a working slot list $S = \emptyset$;
for *each round* $t \in T$ **do**
    $s^\star = null$; // $s^\star$ is the tightest slot of which the smallest empty space is left if the current round t is put in it.
    for $s \in S$ **do**
        find the tightest slot $s^\star$;
    /* there is no slot with enough space to place it */
    if $s^\star = null$ **then**
        create a new slot $s_{|S|+1}$;
        place t in the new slot $s_{|S|+1}$;
        add the new slot $s_{|S|+1}$ to S;
    else
        place t in $s^\star$;
$M = |S|$; // the minimum number of needed working slots
$N = \lceil M/K \rceil$; // the minimum number of needed robots
Return N, S;

The ratio of the optimal value obtained by BFD and MIP is proved to be bounded between $\frac{11}{9}$ and $\frac{5}{4}$.[19] Therefore, it gives very approximate results. The results of robots needed to serve 25 racks for the first eight months obtained by Model B and BFD are compared in Fig. 29.

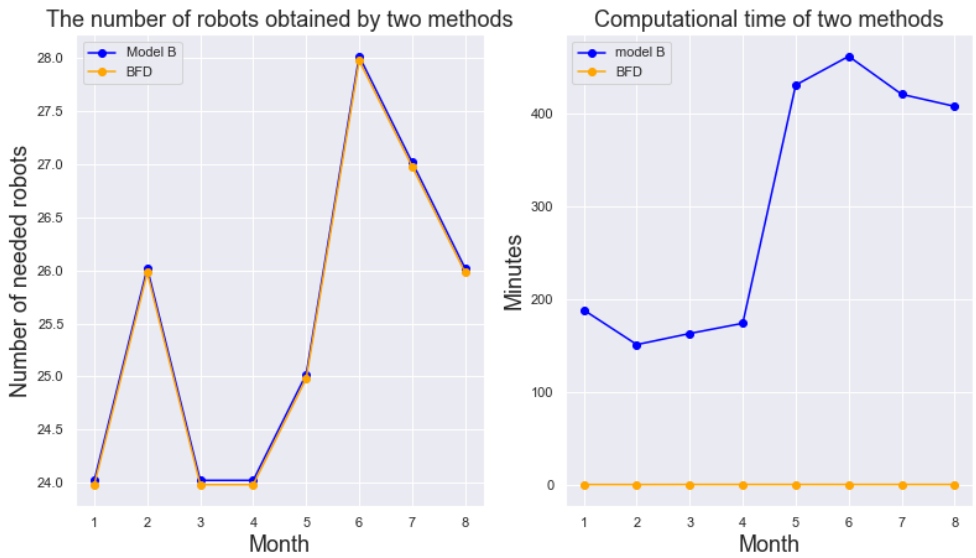


Figure 29: Comparison of the results obtained by Model B and BFD (25 racks)

The first subplot of Fig. 29 displays the number of robots obtained by Model B and BFD respectively, and it can be seen that the two methods give exactly the same solution to the problem. This is due to the fact that although the two methods may get a different total number of needed working slots (M), the number of robots (N) is the rounding up result of the number of working slots (M) divided by the number of working slots of each robot (K), i.e., $N = \lceil M/K \rceil$. This rounding up gap brings benefit to BFD so that it is able to give a more highly approximate result than Model B. The second subplot compares the computational time of the two methods. It takes hours for Model B to provide a solution to the problem but BDF only needs a few seconds. Consequently, BFD is adopted to calculate the minimum number of needed robots for each month after the placements of all racks are determined.

3.5 Optimization of the farm size and the number of racks

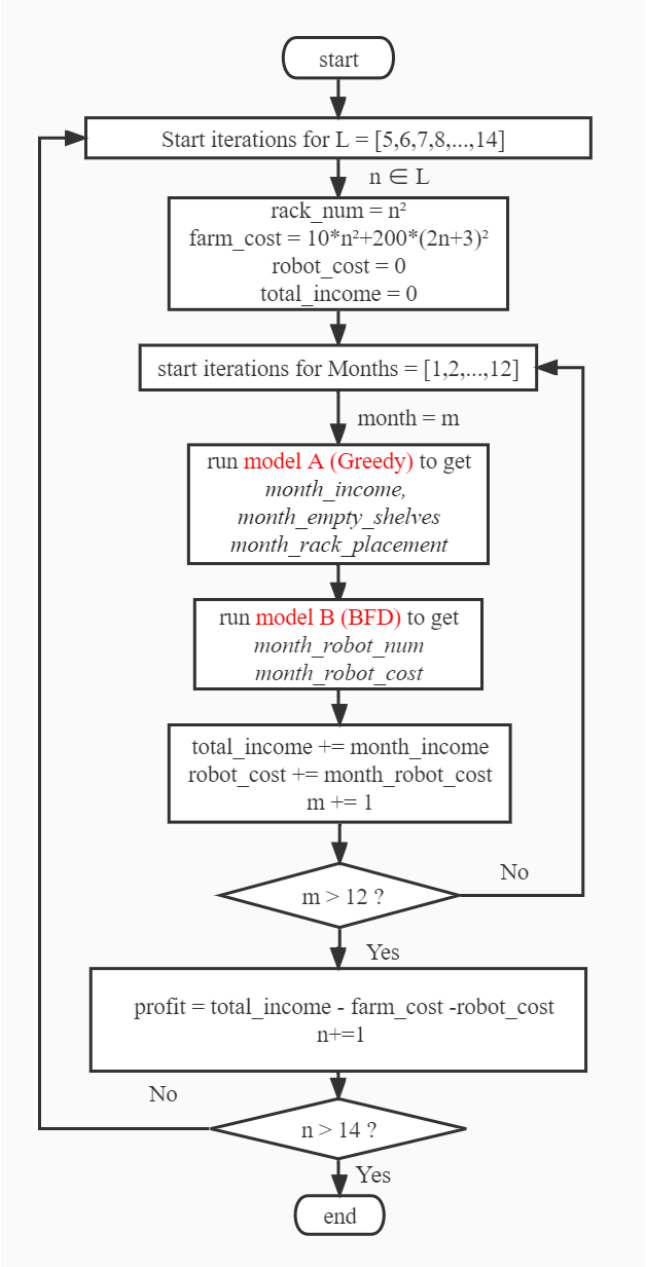


Figure 30: The process of optimizing the farm size and the number of racks

Model A and Greedy algorithm are adopted to solve the placement of all products on all racks in a farm with a fixed number of racks. Model B and Best fit decreasing (BFD) algorithm are used to

calculate the optimal number of needed robots after the placement of all racks is determined by Model A and Greedy algorithm. There is a conflict between the cost of the farm and racks and the cost of robots, because if the farm is small and the number of racks is very limited, then the racks will be full of products with few empty shelves, so it takes longer time for robots to serve all racks, which means the number of robots increases. Therefore, when the farm cost is less, the robot cost will be more and vice-versa. Then, the process of selecting an optimal number of racks and their corresponding farm size can be done as shown in Fig. 30. The main idea is to iterate the number of racks from 5×5 to 14×14 and calculate the corresponding profit, and eventually pick the optimal.

4 Results

In this section, the process in Fig. 30 is executed to study the optimal choice of farm size to make the most profit by gradually increasing the number of racks from 5×5 to 14×14 in the first place. Next, some farms with a different number of racks are picked to research how much profit they will miss each month. Finally, the results will be discussed.

4.1 Scenario 1: Optimal farm size with the number of racks

The number of racks is increased from 5×5 to 14×14 in this scenario.

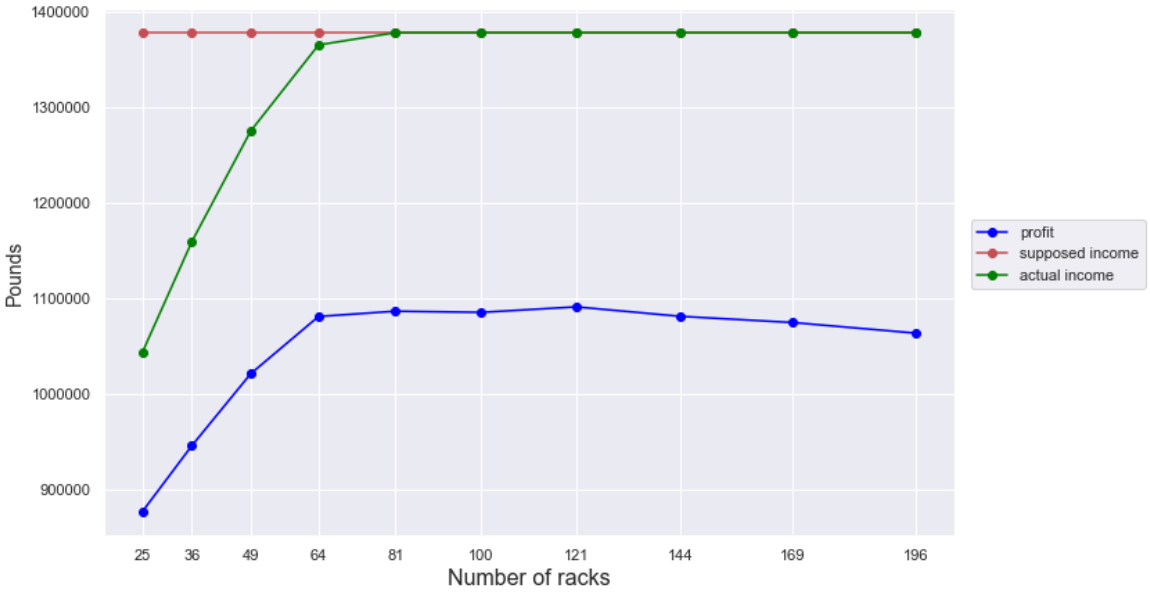


Figure 31: The trend of profit and actual income as the number of racks increases[1]

Fig. 31 displays that, as the number of racks increases, the actual income gradually boosts and reaches the supposed income when there are 9×9 racks, and then remains the same. It means that racks are not enough for all products at the initial stage and more and more products can be planted when there are more racks so the actual income is increasing. 9×9 racks are the point when the number of racks becomes sufficient for the demand of all products. Meanwhile, the profit also rises significantly when the actual income boosts, but the rate of growth becomes milder after 8×8 racks, then eventually reaches the maximum at 11×11 racks and drops after that.

[1]The supposed income is a fixed value which describes the income of the demands of all products being met, while the actual income is what the products being placed in racks can earn.

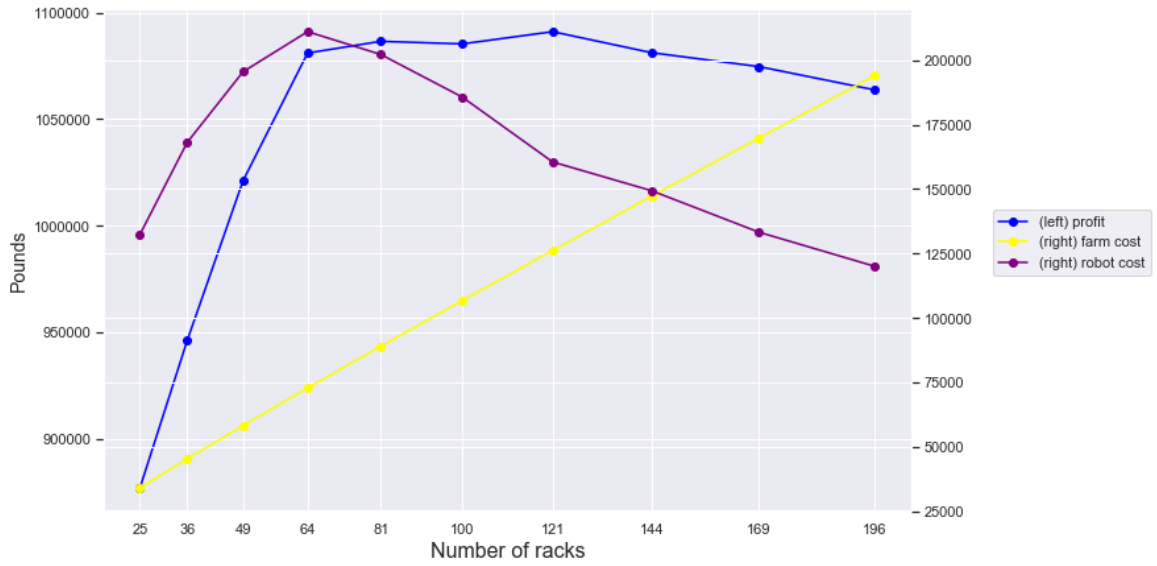


Figure 32: The trend of profit, farm cost and robot cost as the number of racks increases

Fig. 32 indicates that, as the number of racks increases, the farm cost rises correspondingly, but the robot cost increases before it reaches 8×8 racks and then drops after that. This is because when the number of racks increases from 5×5 to 8×8, there are more and more products being placed in racks, so the demand for robots gets higher. However, if there are more than 8×8, the products being placed in racks are fixed, so there are more and more empty shelves in all racks as shown in Fig. 33 and the demand for robots becomes lower.

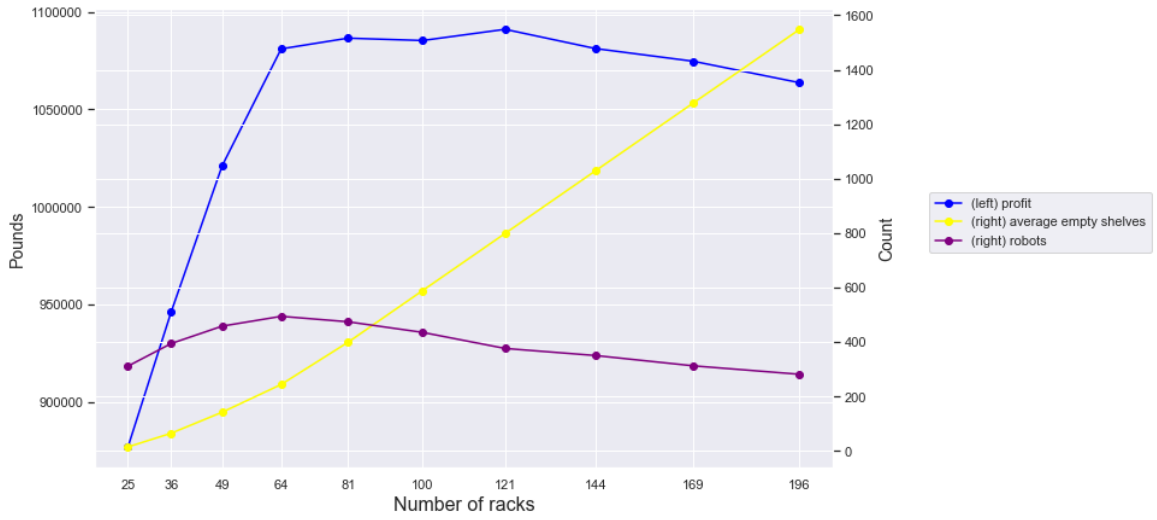


Figure 33: The trend of profit, average empty shelves and robots as the number of racks increases

4.2 Scenario 2: Behaviors of some farm sizes

In this scenario, four different farms with the number of racks being 5×5, 6×6, 7×7, and 8×8 are generated respectively to compare their monthly behaviors.

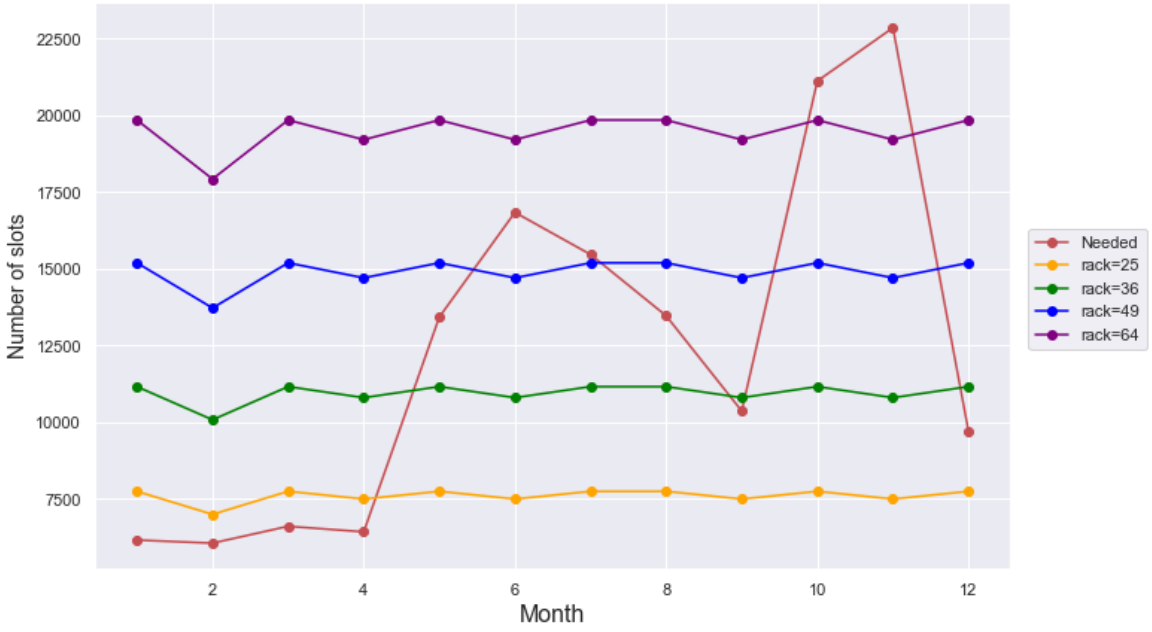


Figure 34: The monthly needed slots and available slots of different farms[1]

The red line in Fig. 34 tells the number of monthly needed slots if the demand for all products is satisfied. The rest four lines show the number of available slots of every farm each month. It can be told that for the first four months, all generated farms have sufficient slots to plant all required products. However, for the rest months, for instance, in the fifth month, only the farms with 7×7 racks and 8×8 racks get enough slots for them, and the demand for some products cannot be satisfied by the farms with 5×5 racks and 6×6 racks. The same interpretation can be used for the rest of the months.

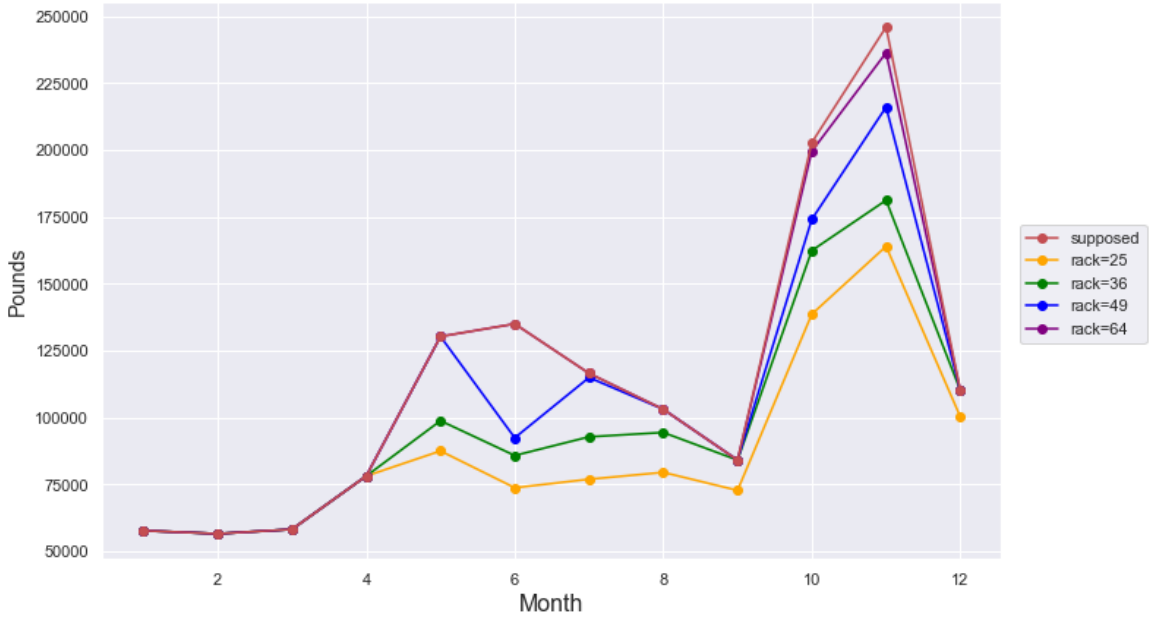


Figure 35: The monthly income of different farms

Fig. 35 displays the monthly income of different farms, and the actual income gets closer to the supposed income when there are more racks because more products are planted.

[1]The available slots of a month is defined as *shelf number* $\times$ *the number of days in the month*. For example, 25 racks have $25 \times 10 \times 31$ slots in January.

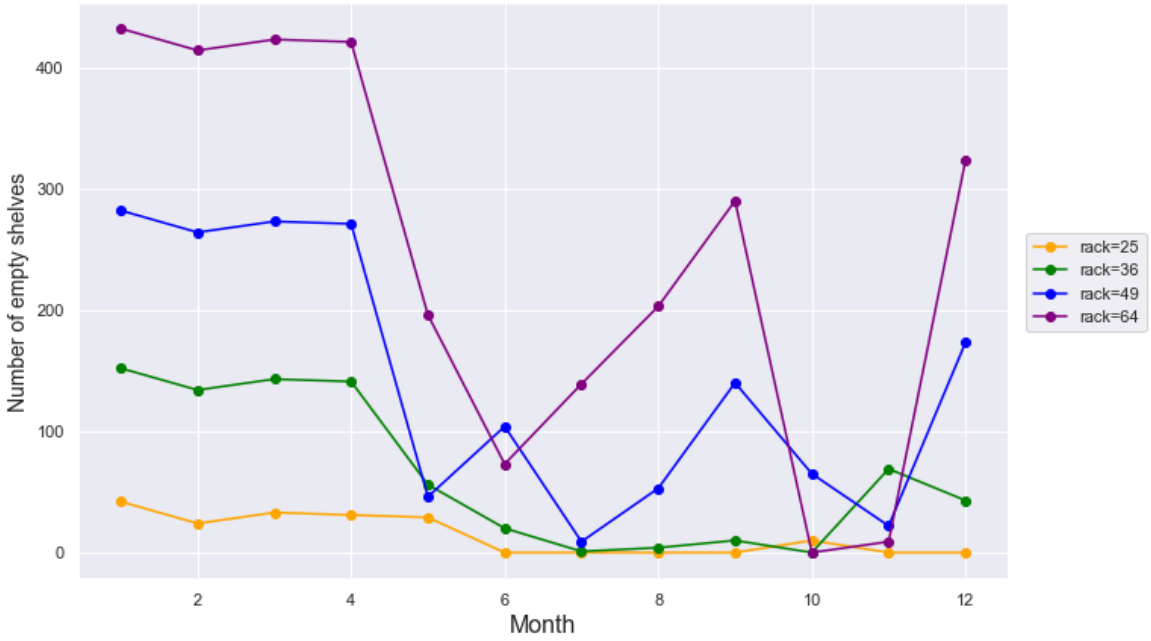


Figure 36: The monthly empty shelves of different farms

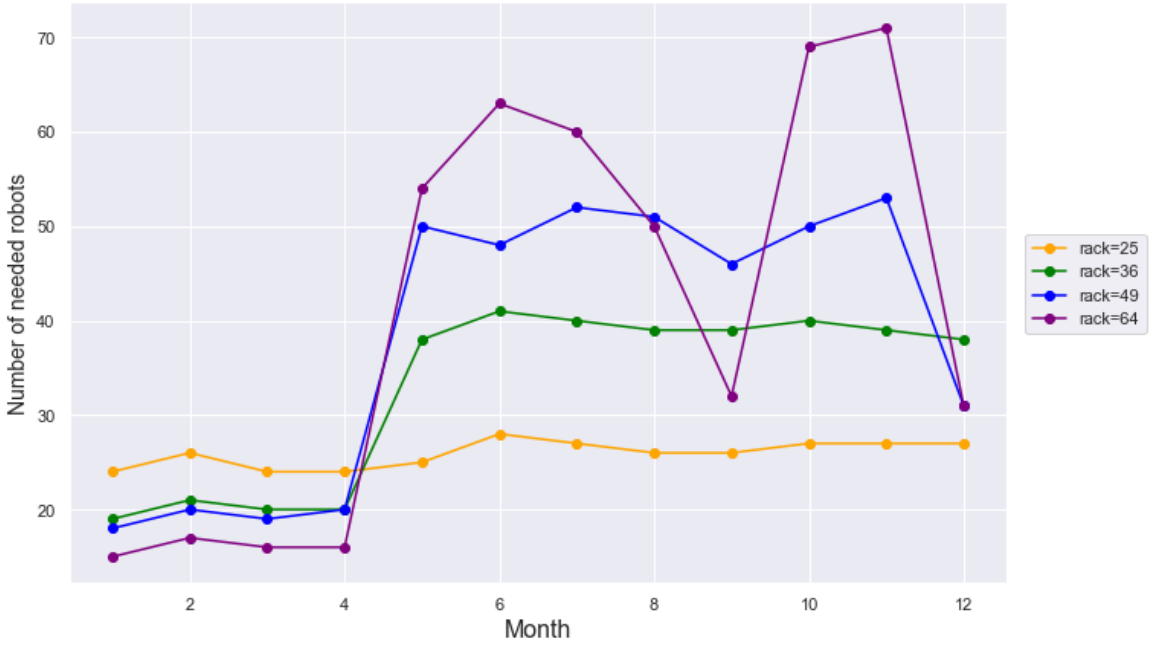


Figure 37: The monthly needed robots of different farms

Fig. 36 and Fig. 37 demonstrate the monthly empty shelves and needed robots of different farms. The two plots can be interpreted as follows. For the first four months, since the available slots are enough for the demand, the more racks there are on the farm, the more monthly empty shelves, and the fewer needed robots. However, when the available slots are not enough, for example in the fifth month, there exist some situations where a farm with 7×7 racks has fewer empty shelves and needs more robots than that with 6×6 months because more products are planted in the former farm. This is also the situation for the rest of the months.

4.3 Discussion

When it comes to solving problems with MIP models, the most important thing is to have a good understanding of the purpose of the problem and make certain reasonable assumptions. Before building each model, it is essential to sort out all known parameters, and the required constraints in the first place. Setting appropriate variables is the next stage in creating a model that can meet these requirements. In vertical farming, the problems are usually very complicated and it is almost impossible to put all elements in just one MIP model to find a solution. Therefore, two or more models need to be constructed to solve different subproblems, and a connection between the models is required to give a final solution. In addition, the speed of running each model needs to be taken into account. If, as in this vertical farm problem, each model needs to be run hundreds of times to find the optimal solution, then optimizing the speed at which each model runs becomes crucial. In this case, the loss of results in Greedy algorithm compared with Model A is at most 8% but the time difference is significantly huge. Greedy algorithm only takes a few seconds and remains stable regardless of the number of racks but Model A takes hours and even runs out of memory as the farm size increases. BFD performs very well and gives optimum solutions in test samples, and only takes a few seconds while model B needs several hours. With such slight losses and a huge difference in speed, it is vital to create these heuristic algorithms to speed up the model.

The above results depict the behaviors of different farm sizes in two scenarios. As can be seen from Figure 31, the choice of 11×11 racks maximizes profitability in this study. This is an overall evaluation of the following three considerations, the proportion of product demand is met, the farm size, and the number of robots. For this research, the income of the products dominates the farm cost and robot cost, and therefore it is better to fulfill the demand of all products and this decides that the racks have to be at least 9×9. The next step is to weigh up the farm cost against the robot cost, but both of these have a much smaller impact on profit than the product income, which explains why the profit growth becomes smaller between 8×8 racks and 11×11 racks. After reaching its maximum at 11×11, the profit starts to drop because the reduction in robot costs will not cover the increase in farm cost any longer. Consequently, it is recommended to pick the farm of 11×11 racks.

The data set and assumptions of this study also had significant effects on the results obtained. In this research, the revenue generated by the product itself is much greater than the farm cost and robot cost, so the results prefer to choose a farm that satisfies the demand for all products in the first place and then balance the impact of the farm cost and the robot cost. At the same time, the assumptions of this study may result in the number of robots obtained by Model B being less than the number of robots actually needed if the farm is of a very large size and the horizontal movement of robots cannot be eliminated.

This result can give some inspiration for the cultivation of vertical farms. Since the cost of growing a vertical farm depends mainly on the farm cost and the serving cost, and the farm cost may get very expensive if it is located in the center of popular cities, then traditional products with low retail prices are not suitable for vertical farms. Thus, a suggestion is to grow those products with a relatively high sell price but a certain level of demands. Another recommendation is that one should try to make the product income dominates the farm cost and serving cost as much as possible as shown in this example.

Meanwhile, the result will change according to different circumstances. Since it can be assumed that the sell price of products is relatively high and the price of robot remains stable regardless of locations, the factors that may vary much and affect the result are mainly the rent price of a farm and the demand of products.

The rent price of a farm may get very high in the center of some popular cities. In this case, the farm cost will conflict with the income of the product because the product income does not far exceed the farm cost any more, then abandoning some product demands would be a better option compared with fulfilling all of them to pick a farm size producing the maximum profit.

There is another situation which is the exact opposite of the one previously described, and that is the rent price of a farm may get very low in some towns. This situation would be that the robot cost would far exceed the farm cost at the same time as the product income dominates the robot cost. Firstly, it is necessary to meet the demand for all products to achieve maximum revenue. The next

step is to increase the area of the farm as much as possible so that the number of bots is small. The most ideal situation is that this farm would go from being a vertical farm to a normal farm, which means each rack only takes one tray of product on the first shelf and leaves the rest shelves empty. However, it would contradict the previous assumption, which eliminates the horizontal moving time of the robots because too few products are planted on each rack, the vertical moving time of robots decreases significantly and one robot will finish one rack much quicker. Conversely, their horizontal moving time increases dramatically because when a robot finishes one rack it will start another one, so the number of racks one robot serves increases. Therefore, when the farm area expands to a certain value, the robot's horizontal moving time should be taken into account. Then the corresponding model needs to be reconsidered.

In terms of the product demand, in this research, the demand for products is relatively small and only one farm is required. If there is a high demand for products, it is needed to be discussed on a case-by-case basis as previously described. An initial step is to consider whether there is a significant difference between the product revenue and the farm cost. If the two are on a similar level, then a balance needs to be found and some products may need to be dropped at this point. If the product revenue dominates the farm cost, the demand for all products needs to be met. The following two measures can be taken at this time. The first is to increase the size of the farm, which brings back to the previous discussion. Secondly, one can increase the number of farms by renting more floors in the same building for multiple farms. The question then becomes how the product should be distributed to each farm, after which the operation of each farm can still be implemented in the way given in the thesis.

5 Conclusions and future research

This study develops an approach to the problem of how one could run a sustainable indoor vertical farm.

The first chapter introduces the concept of the vertical farm and the reasons for its emergence. Next, some of the advantages of vertical farms compared to traditional farms are described, and their current shortcomings are presented. A general overview of the current situation of vertical farms is provided.

The second chapter describes the specific problem of this thesis and poses three research questions. This is followed by a list of relevant previous research and informs what improvements have been made in this paper. Besides, some prior knowledge is added to make it easier for readers to understand the following chapters.

The third chapter presents the methodology and logic behind it. Importantly, two mixed-integer linear programming (MIP) models are presented in this section to solve the problem of monthly rack placement and optimization of the number of robots. In addition, as the models run slowly, two heuristic algorithms are also proposed to improve the speed. The three research questions were answered properly in Section 3.1,3.2 and 3.5.

In the fourth chapter, the results show that the two applied models with their heuristics perform effectively to find the optimal solution to the problem. Two scenarios are presented to find the optimal farm size with the number of racks and explore the behaviors of some farm sizes respectively. A detailed discussion is followed to find out what measures should be taken in different situations.

These findings have significant implications for the understanding of how to apply MIP models in the field of vertical farming. (i) The original problem was converted into two MIP models to solve the two problems of product placement on racks and how robots should serve all racks respectively. (ii) Two heuristics are proposed to improve the efficiency of solving the two MIP models. (iii) There is a good connection between the two models. The first MIP model (model A) simplifies the second problem in Section 3.4 by allowing it to have the first assumption because of the high similarity of water frequencies on all racks.

For future work, it will be interesting to investigate the following four issues, robot's horizontal moving time, robot's movement route, the positions of charging stations and the price fluctuation of products.

- Robot's horizontal moving time. How the second model (Model B) can be modified if its first assumption is removed, i.e., the horizontal moving time of each robot cannot be eliminated, when the farm gets extremely large. This may turn the problem into a non-trivial Vehicle Routing Problem as in [16].

- Robot's movement route. This research only leaves sufficient time for robots to move but does not take the route into account. How to plan the trajectory of these robots so that they can reach their destination within the specified time and they do not conflict with each other while moving is also a topic worth investigating.

- Positions of charging stations. This study assumes that there are charging points near each rack, enabling the robots to be charged in time. In fact, it is also worth studying where the charging stations should be placed on the farm.

- Price fluctuation of products. In this research, the retail price in the data set takes the average value and remains fixed for the whole year, so it is worth thinking about how the approach should be changed if the price fluctuation of the products is taken into account.

References

[1] Examples of exponential smoothing methods. https://www.statsmodels.org/v0.10.2/examples/notebooks/generated/exponential_smoothing.html. Accessed: 2022-08-15.

[2] Quarterly visitor nights spent by international tourists to australia 1999-2010.

[3] Chapter 12 seasonal autoregressive integrated moving average models. In K. W. Hipel and A. I. McLeod, editors, *Time Series Modelling of Water Resources and Environmental Systems*, volume 45 of *Developments in Water Science*, pages 419–462. Elsevier, 1994.

[4] *Kolmogorov–Smirnov Test*, pages 283–287. Springer New York, New York, NY, 2008.

[5] K. S. Abukhader R. Artificial intelligence for vertical farming. mälardalen university; 2021. master's thesis.

[6] B. Adenso-Díaz and G. Villa. Crop planning in synchronized crop-demand scenarios: A biobjective optimization formulation. *Horticulturae*, 7(10), 2021.

[7] K. Al-Kodmany. The vertical farm: A review of developments and implications for the vertical city. *Buildings*, 8(2), 2018.

[8] Alex. How to do cross-validation in time series prediction? https://zhuanlan.zhihu.com/p/141542218. Accessed: 2022-06-28.

[9] S. Angel, J. Parent, D. L. Civco, A. Blei, and D. Potere. The dimensions of global urban expansion: Estimates and projections for all countries, 2000–2050. *Progress in Planning*, 75(2):53–107, 2011. The dimensions of global urban expansion: Estimates and projections for all countries, 2000–2050.

[10] K. Benke and B. Tomkins. Future food-production systems: vertical farming and controlled-environment agriculture. *Sustainability: Science, Practice and Policy*, 13(1):13–26, 2017.

[11] B. Billah, M. King, R. Snyder, and A. Koehler. Exponential smoothing model selection for forecasting. *International Journal of Forecasting*, 22:239–247, 02 2006.

[12] G. E. P. Box and D. A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, 65(332):1509–1526, 1970.

[13] Y.-W. Cheung and K. S. Lai. Lag order and critical values of the augmented dickey–fuller test. *Journal of Business & Economic Statistics*, 13(3):277–280, 1995.

[14] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

[15] Dake. Illustration of the knapsack problem. which boxes to choose to maximize the amount of money while still fullfilling the 15 kg constraint ? https://commons.wikimedia.org/wiki/File:Knapsack.svg. Accessed: 2022-08-15.

[16] M. Delorme and A. Santini. Energy-efficient automated vertical farms. *Omega*, 109:102611, 2022.

[17] N. V. Fedoroff. Food in a future of 10 billion. *Agriculture & Food Security*, 4(1):1–10, 2015.

[18] R. Fried and A. C. George. *Exponential and Holt-Winters Smoothing*, pages 488 – 490. 06 2009.

[19] M. R. Garey, R. L. Graham, and J. D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 143–150, 1972.

[20] M. R. Garey and D. S. Johnson. A guide to the theory of np-completeness. 1978.

[21] N. Gröwe-Kuska and W. Römisch. *Stochastic unit commitment in hydro-thermal power production planning*. Preprints aus dem Institut für Mathematik. Humboldt-Universität zu Berlin, Institut für Mathematik, 2001.

[22] H. H. Hoos and E. Tsang. Chapter 5 - local search methods. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 135–167. Elsevier, 2006.

[23] R. Hyndman and Y. Khandakar. Automatic time series forecasting: the forecast package for r. *Journal of Statistical Software*, 26(3):1–22, 2008.

[24] T. S. Jayne, J. Chamberlin, and D. D. Headey. Land pressures, the evolution of farming systems, and development strategies in africa: A synthesis. *Food policy*, 48:1–17, 2014.

[25] D. Jungnickel. *The Greedy Algorithm*, pages 129–153. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[26] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. 01 2004.

[27] R. E. Korf. An improved algorithm for optimal bin packing. In *IJCAI*, volume 3, pages 1252–1258. Citeseer, 2003.

[28] W. Lutz and S. Kc. Dimensions of global population projections: What do we know about future population trends and structures? *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 365:2779–91, 09 2010.

[29] Martello, Silvano, and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[30] B. R. B. Mohamed N. Nounou. *Multiscale Methods for Denoising and Compression*. 2000.

[31] S. Noor, M. I. Lali, and M. S. Nawaz. Solving job shop scheduling problem with genetic algorithm. *Sci. Int.(Lahore)*, 27(4):3367–3371, 2015.

[32] U. Obu, G. Sarkarkar, and Y. Ambekar. Computer vision for monitor and control of vertical farms using machine learning methods. In *2021 International Conference on Computational Intelligence and Computing Applications (ICCICA)*, pages 1–6, 2021.

[33] P. O'Neill, D. Nicolaides, D. Honnery, and J. Soria. Autocorrelation functions and the determination of integral length with reference to experimental and numerical data. 01 2004.

[34] M. H. M. Saad, N. M. Hamdan, and M. R. Sarker. State of the art of urban smart vertical farming automation system: Advanced topologies, issues and recommendations. *Electronics*, 10(12), 2021.

[35] S. Sankaran. Demand forecasting of fresh vegetable product by seasonal arima model. *Int. J. of Operational Research*, 20:315 – 330, 01 2014.

[36] T. Shiina and J. R. Birge. Stochastic unit commitment problem. *International Transactions in Operational Research*, 11(1):19–32, 2004.

[37] L. Sun, X. Cheng, and Y. Liang. Solving job shop scheduling problem using genetic algorithm with penalty function. *International Journal of Intelligent information processing*, 1(2):65–77, 2010.

[38] P. (talk). Comparison of two simulated processes, one stationary, one nonstationary. `https://commons.wikimedia.org/wiki/File:Stationarycomparison.png`. Accessed: 2022-08-15.

[39] S. Varma and R. Simon. Bias in error estimation when using cross-validation for model selection. *BMC bioinformatics*, 7(1):1–8, 2006.

[40] Y. Wang, X. Chang, M. Gao, and X. Hou. Seasonal autoregressive integrated moving average model for precipitation time series. *Journal of Mathematics and Statistics*, 8:500–505, Feb 2013.

[41] W. Wei. *Time Series Analysis: Univariate and Multivariate Methods*, volume 33. 01 1989.

[42] C.-L. Yang, Y. Hari, and Y.-F. Kuo. Multiple-crop scheduling for plant factory. *Ismab'12*, pages 18–20, 2012.

[43] C.-L. Yang, S.-J. Huang, and C.-H. Ang. Recursive heuristic scheduling method for multi-crop plant factory with solar panel roof. *Computers and Electronics in Agriculture*, 165:104941, 2019.

Appendices

Rack/ Shelf	1	2	3	4	5	6	7	8	9	10
1	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)		
2	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
3	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)		
4	strawberries (11)	asparagus (10)	oranges (8)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)		
5	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
6	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	green beans (7)	basil (6)	basil (6)	basil (6)		
7	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
8	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	green beans (7)	basil (6)	basil (6)	basil (6)		
9	strawberries (11)	asparagus (10)	oranges (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
10	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
11	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
12	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)		
13	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
14	strawberries (11)	asparagus (10)	mushrooms (8)	oranges (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)		
15	strawberries (11)	mushrooms (8)	mushrooms (8)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)		
16	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)		
17	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
18	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
19	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
20	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	green beans (7)	basil (6)	basil (6)	basil (6)		
21	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
22	strawberries (11)	chestnuts (9)	mushrooms (8)	oranges (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
23	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		
24	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)		
25	strawberries (11)	asparagus (10)	mushrooms (8)	mushrooms (8)	chicory (7)	basil (6)	basil (6)	basil (6)		

Table 4: Example of the placement of 25 racks in January obtained by Model A

Rack/ Shelf	1	2	3	4	5	6	7	8	9	10
1	strawberries (11)	asparagus (10)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)	basil (6)	basil (6)	
2	strawberries (11)	asparagus (10)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)	basil (6)	basil (6)	
3	strawberries (11)	asparagus (10)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)	basil (6)	basil (6)	
4	strawberries (11)	asparagus (10)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)	basil (6)	basil (6)	
5	strawberries (11)	asparagus (10)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)	basil (6)	basil (6)	
6	strawberries (11)	asparagus (10)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)	basil (6)	basil (6)	
7	strawberries (11)	asparagus (10)	mushrooms (8)	winter squash (7)	basil (6)	basil (6)	basil (6)	basil (6)	basil (6)	
8	strawberries (11)	asparagus (10)	mushrooms (8)	winter squash (7)	basil (6)	basil (6)	basil (6)	basil (6)	basil (6)	
9	strawberries (11)	spring greens (10)	mushrooms (8)	winter squash (7)	basil (6)	basil (6)	basil (6)	basil (6)		
10	strawberries (11)	spring onions (8)	mushrooms (8)	green beans (7)	basil (6)	basil (6)	basil (6)	basil (6)		
11	strawberries (11)	spring onions (8)	mushrooms (8)	green beans (7)	basil (6)	basil (6)	basil (6)	basil (6)		
12	strawberries (11)	spring onions (8)	mushrooms (8)	green beans (7)	basil (6)	basil (6)	basil (6)	basil (6)		
13	strawberries (11)	spring onions (8)	mushrooms (8)	green beans (7)	basil (6)	basil (6)	basil (6)	basil (6)		
14	strawberries (11)	spring onions (8)	mushrooms (8)	green beans (7)	basil (6)	basil (6)	basil (6)	basil (6)		
15	strawberries (11)	spring onions (8)	mushrooms (8)	green beans (7)	basil (6)	basil (6)	basil (6)	basil (6)		
16	asparagus (10)	spring onions (8)	mushrooms (8)	green beans (7)	basil (6)	basil (6)	basil (6)	basil (6)		
17	asparagus (10)	spring onions (8)	oranges (8)	green beans (7)	basil (6)	basil (6)	basil (6)	basil (6)		
18	asparagus (10)	spring onions (8)	swiss chard (7)	green beans (7)	basil (6)	basil (6)	basil (6)	basil (6)		
19	asparagus (10)	mushrooms (8)	swiss chard (7)	green beans (7)	basil (6)	basil (6)	basil (6)	basil (6)		
20	asparagus (10)	mushrooms (8)	swiss chard (7)	chicory (7)	basil (6)	basil (6)	basil (6)	basil (6)		
21	asparagus (10)	mushrooms (8)	swiss chard (7)	chicory (7)	basil (6)	basil (6)	basil (6)	basil (6)		
22	asparagus (10)	mushrooms (8)	swiss chard (7)	chicory (7)	basil (6)	basil (6)	basil (6)	basil (6)		
23	asparagus (10)	mushrooms (8)	swiss chard (7)	chicory (7)	basil (6)	basil (6)	basil (6)	basil (6)		
24	asparagus (10)	mushrooms (8)	swiss chard (7)	chicory (7)	basil (6)	basil (6)	basil (6)	basil (6)		
25	asparagus (10)	mushrooms (8)	swiss chard (7)	basil (6)	basil (6)	basil (6)	basil (6)	basil (6)		

Table 5: Example of the placement of 25 racks in January obtained by Greedy algorithm