Computers & Industrial Engineering 157 (2021) 107263

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Modeling and optimization of watering robot optimal path for ornamental plant care

Maoqing Zhang [a], Weian Guo [b,c,*], Lei Wang [a,c], Dongyang Li [a], Bo Hu [a], Qidi Wu [a]

[a] School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China
[b] Sino-Germany College of Applied Sciences, Tongji University, Shanghai 201804, China
[c] Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai 200092, China

ARTICLE INFO

Keywords:
Curse of dimensionality
Insufficient irrigation
Placeholder strategy
Sliding window strategy
Genetic algorithm

ABSTRACT

Watering ornamental plants is considered to be a time-consuming task with the increasing garden size. Employing robots to conduct the watering task is a feasible way to improve the watering efficiency. However, general path planning models for watering robots are faced with various difficulties, such as the curse of dimensionality caused by the excessively oversized irrigation area and insufficient irrigation due to weather conditions. To tackle these issues, this paper proposes two novel strategies for watering robot optimal path modeling. Firstly, for tackling the curse of dimensionality, a sliding window strategy is proposed. To be specific, the whole watering area is partitioned into many subareas using the sliding window, and each subarea is watered independently. Secondly, to overcome the problem that the soil moisture still can not reach the expectation due to various weather conditions, this paper proposes a placeholder strategy, which enables the watering robot to dynamically adjust the watering path. To test these strategies, a novel genetic algorithm with neighbor exchanging strategy is proposed. Extensive experiments demonstrate the effectiveness of watering robot path planning models based on the proposed strategies.

1. Introduction

Ornamental plants, such as cymbidium, brazilwood, aloe vera and areca palm, are common indoor plants, which are often placed in conference rooms, hotel lobbies and leisure squares. However, with the increasing garden size, it is considered to be a time-consuming and laborious task to complete the whole watering process. Common sprinkler and drip irrigation systems are applicable for gardens with a single variety of flowers, but not suitable for gardens with various watering requirements. In terms of various difficulties and corresponding solutions in common irrigation systems, they can be summarized as follows.

Some works are focused on the watering process. A system (Azhar, Irawan, & Saputra, 2017) is designed based on Internet of Things(IoT) integrated with mobile. In India, farmers are constantly troubled with the problem that the power is frequently cutted off, resulting in the high wastage of water resources. To tackle this issue, this system employs temperature and moisture sensors integrated with raspberry-Pi to remotely control the water pump. To properly water the garden, one system (Shawn, 2014) is developed based on weather forecast and timer. The system can automatically look up the forecast using predefined API and conduct watering actions. In addition, past, current and forecasted weather conditions can also be included into the system to make a proper watering plan for plants.

Some works are focused on the evaluation of water demands. A smart controlling system (Gubbi, Buyya, Marusic, & Palaniswami, 2013) is developed based on the wireless sensor work and cloud computing to measure the watering amount. In this system, Zigbee is used as a bridge between the sensors nodes and base stations. Besides, actuators are employed for measuring the water demand of plants. A GSM module (Pavithra & Srinath, 2014) is embedded into intelligent monitoring system to evaluate the water-level tank, and based on the evaluated figure, the exact amount of water need is provided. In addition to that, humidity and temperature can also be evaluated for plant growth.

Smartphone-based applications can not only help owners of plants to remotely observe the conditions of plants, but also conduct necessary watering actions. A user friendly application (Azhar et al., 2017) is designed to serve as the bridge between plant owners and remote plants. Using the application, users can have a better understanding of the remaining percentage of water tank, air humidity and temperature. The

* Corresponding author.
E-mail addresses: maoqing_zhang@163.com (M. Zhang), guoweian@163.com (W. Guo), wanglei@tongji.edu.cn (L. Wang).

https://doi.org/10.1016/j.cie.2021.107263
Received 24 October 2020; Received in revised form 14 January 2021; Accepted 21 March 2021
Available online 25 March 2021

application allows users to set different parameters to satisfy the needs of various plants. An intelligent monitoring platform framework and system structure for facility agriculture ecosystem based on Internet of Things is provided (Ram, Vishal, Dhanalakshmi, & Vidya, 2015). In China, the facility agriculture area is expanding. However, the lack of exact information and communication leads to the loss in production. The developed platform can be a catalyst for the transition from traditional farming to modern farming. This also provides opportunities for creating a new technology and service development in IoT farming application. Network safety is always a concern in IoT, and it is also critical for remote control of watering systems. Blockchain (Hua, Wang, Kang, Wang, & Wang, 2018), which is well known in recent years for its traceability, is applied for securing data transmission in watering systems. An agricultural provenance system (Hua et al., 2018) is designed in combination with the blockchain technique. Information, such as fertilizing and irrigation, are recorded to solve the trust crisis in product supply. The application of blockchain builds a reliable community among different stakeholders (growers, farmers and sellers).

Optimization of decision results is another aspect of intelligent watering systems. Generally, environmental parameters, such temperature, humidity, and water content in soil, are input into the system to make an optimal decision. However, due to the existence of sensor noise, Kalman filter is used in the system (Pienaar, Fisher, & Hancke, 2015) to decrease the effect of the noise on the final decision. Sharma, Shinde, and Chougle (2015) propose to implant sensors into soil for measuring moisture level and checking water level in tank. Intelligent softwares on servers are used to analyze the sensed data, and the effective decisions are made to take proper watering actions.

These methods above are effective in providing watering solutions to different situations. However, when it comes to simultaneously watering various plants in the same place, these methods are infeasible due to the lack of considerations of multiple constraints during the watering process. For example, there are different kinds of plants in one garden, and these plants may have various watering requirements. Some plants may need to be watered many times according to their living habits, while other plants may be irrigated only once. Obviously, common drip irrigation systems are unable to satisfy the individualized requirements. In addition, with the increasing amount of plants, the watering efficiency also becomes a serious problem. Therefore, an efficient watering robot path planning model is of great necessity. Essentially, this problem is a robot path planning problem, but it is evidently different from regular path planning problems due to the existence of various watering needs. Thus, to tackle this issue, this paper firstly analyzes a general and basic watering model; After that, to improve the basic watering model, an extended watering model is firstly proposed, which incorporates a sliding window strategy into the basic watering model. Considering the dynamical watering requirement in special weather, this paper further proposes a placeholder strategy and incorporates it into the extended watering model. Additionally, a novel genetic algorithm is specially designed to tackle the watering models above, Therefore, the contributions of this paper can be summarized as follows:

(1) A new watering model, termed as extended watering model, is proposed by incorporating the sliding window strategy. Firstly, to have a intuitionistic understanding of the curse of dimensionality in modeling the watering path, this paper presents a general and basic watering model. The performance of the basic watering model dramatically deteriorates with the increasing garden size according to the detailed analyses. To tackle this problem, this paper further utilizes a sliding window strategy to divide one garden into multiple small watering regions, and each region is optimized individually.
(2) A dynamic watering model is proposed to adapt to dynamic watering requirements. In hot weather, although plants have been irrigated according to their requirements, the soil moisture is still unable to reach the standard. Therefore, the watering robot is required to dynamically adjust the watering path. To tackle this issue, a dynamic extended watering model is proposed for various possible watering scenarios, which mainly incorporates a placeholder strategy into the extended watering model.
(3) The two proposed watering models are tested in detail using a specially designed genetic algorithm. To optimize the proposed watering models, this paper specially designs an optimizer based on genetic algorithm, including population initialization strategy, crossover operator and mutation operator. The extended watering model is analyzed and evaluated using the novel genetic algorithm above. Thereafter, the dynamic extended watering model is systematically tested. Experimental results and extensive analysis indicate that the extended watering model is able to adapt to different garden sizes. In addition, further experimental results indicate that the dynamical extended watering model is also able to dynamically adjust watering path according to various requirements during watering process, as well as achieve a high time saving rate under various test conditions.

The structure of this paper is organized as follows. In Section 2, related work on robot path planning is presented. Following that, two watering models are articulated, as well as the phenomenon of the curse of dimensionality in Section 3. A novel genetic algorithm is presented in Section 4. Extensive experiments and results analysis are conducted in Section 5, where three experiments are conducted to fully investigate the designed watering models. Section 6 concludes this paper and gives the further research directions.

2. Related work

Robot path planning is an important research field, and it can be applied in many scenarios, such as surveillance, search and rescue, and border patrol. Many methods (Dijkstra, 1959; Hart, Nilsson, & Bertram, 1968; Anthony, 1994; Gerke, 1999) are available to tackle this problem, and they can be roughly divided into two categories: exact methods and heuristic methods. Exact methods include Dijkstra's algorithm (Dijkstra, 1959), A* algorithm (Hart et al., 1968), and D* algorithm (Anthony, 1994), all of which are able to find the optimal solution, but have the drawback of high complexities (Borges, Miyazawa, Schouery, & Xavier, 2020). Heuristic algorithms can obtain a suboptimal solution with low costs. Genetic algorithm (Gerke, 1999), colony optimization algorithm (Dorigo, Maniezzo, & Colorni, 1996), probabilistic roadmap method (Kavraki & Latombe, 1994;Cui et al., 2020; Cui et al., 2020;Cai, Geng, Wu, Cai, & Chen, 2020) are common tools. However, in general, these methods can not be directly applied to robot path planning problems without considering the features of practical problems. According to problem features, recent work on robot path planning is summarized as follows.

Early attempt to tackle mobile robot path planning on terrain maps is made by Gaw and Meystel (1986), who represent the terrain as polygonised isolines. Then, minimum-time trajectories of motion on these maps are calculated using the elevation changes between adjacent isolines. To find energy-efficient paths on uneven terrains, Nuwan, Cheng, and Tse (2015) establish an energy-cost model for mobile robots. The terrains are represented as grid-based elevation maps. By using zigzag-like path patterns, A*-like heuristic search algorithm is used to estimate heuristic energy-costs on steep terrains, which cannot be estimated using traditional methods. Further, Plonski, Tokekar, and Isler (2013) take into consideration energy-efficient time-constrained path planning of a solar-powered robot navigating on uneven terrains. They obtain a solar map using Gaussian process regression. Then, the energy-efficient paths are found based on this map and an empirical model of the robot.

For modern autonomous underwater vehicles, efficient path planning algorithms are a crucial issue. Alvarez, Caiti, and Onken (2004) propose to employ genetic algorithm in combination with dynamic programming. For autonomous underwater vehicles in ocean

environments, the complex spatial variability is likely to jeopardize their missions. To avoid this issue, Garau, Alvarez, and Oliver (2005) propose to plan safety routes with minimum energy cost. In addition, more realistic and applied case of constant thrust power navigation is considered in the method. To efficiently extract a continuous path from a discrete representation of the environment, Petres et al. (2007) develop a novel fast marching based approach, and incorporate underwater currents into the approach.

The categories discussed above are two common cases. Other research on robot path planning can also be found in (Deb & Tiwari, 2008). Due to the limitation of paper research content, more papers on robot path planning are not presented any more. However, to the best of my knowledge, no paper has been reported on the application of robot path planning to the automatic watering problem discussed in this paper. Therefore, to effectively provide a feasible solution to watering robot optimal path, based on the proposed watering models, this paper employs a novel genetic algorithm as a basic solver, and further proposes a new version of genetic algorithm.

3. Modeling of Watering Robot Optimal Path

Watering robot path planning is essentially one component of an automatic watering system. There are also some other components, such as remote control strategies and encryption algorithms. This paper is not intended to introduce the automatic watering system in detail, but briefly illustrates how the system operates for a clear understanding of the watering models proposed in this paper. In terms of the automatic watering system, as Fig. 1 displays, it consists of software and hardware platforms (Kumar, Sudip, & Singh, 2019; Chen & Lin, 2019; Kammoun, Hamidouche, Belghith, Nezan, & Nouri, 2018). The hardware platform includes position sensors (the blue symbol on the left of each flowerpot in Fig. 1), soil moisture sensors (the red symbol on the right of each flowerpot in Fig. 1), server for data analysis, and watering robot. The software platform contains a data analysis module for soil moisture sensors, position analysis module for position sensors, path planning module for watering robot, communication module and data encryption module. The software and hardware platforms collaborate to complete the whole watering process, which can be described with Fig. 1.

The operation process can be briefly described as follows. Firstly, position sensors and soil moisture sensors (Yildiz, 2019; Gunathillake, Huang, & Savkin, 2019) are planted into each flowerpot. After that, position sensors periodically send position information to corresponding server, as well as plant category information over the Internet. Based on the received data, the server calculates the relative position of each plant via the position analysis module. At the same time, soil moisture sensors also periodically measure the soil moisture and send the moisture information to the server. The server analyzes soil moisture information using a data analysis module. According to the analysis results, the server decides whether to start the watering program. If the predefined watering condition is reached, the data analysis module on the server further analyzes and calculates the interval time, watering duration and watering frequency based on the plant category, soil category, and soil moisture information. Note that the watering frequency is determined according to expert experience. In principle, a single watering can meet a predetermined water requirement for any plant.However, it cannot precisely control soil moisture because water diffuses in different soils at different rates. Therefore, this paper adopts the method of multiple watering to accurately control the watering process and soil moisture. The computation results (including plant category, plant position, interval time, watering duration and watering frequency) are automatically sent to the watering robot after the calculation is completed. Once receiving the watering instruction, watering robot is required to plan watering path according to constraint conditions. It should be noted that all data transmitted over the Internet is encrypted using a data encryption module to avoid malicious network attacks and information disclosure (Kim & Lee, 2019; Jia, Li, Yin, Guo, & Gu, 2019; Abdi et al., 2019).

3.1. Basic watering model and its drawback analysis

For simplicity, take a generic scenario as an example to illustrate the basic watering model. Suppose there are various ornamental plants, and these plants are randomly placed in a garden of $N \times N$ plants as Fig. 2 displays. Note that the position sensors, soil moisture sensors and server are hidden to highlight the main components. The locations of these plants are determined a prior. The black point at the top-right corner of Fig. 2 is a watering robot, which is responsible for watering the ornamental plants. Since these plants have different living habits, the watering duration and interval time vary with different plants. To meet different requirements of various plants, and concurrently reduce the

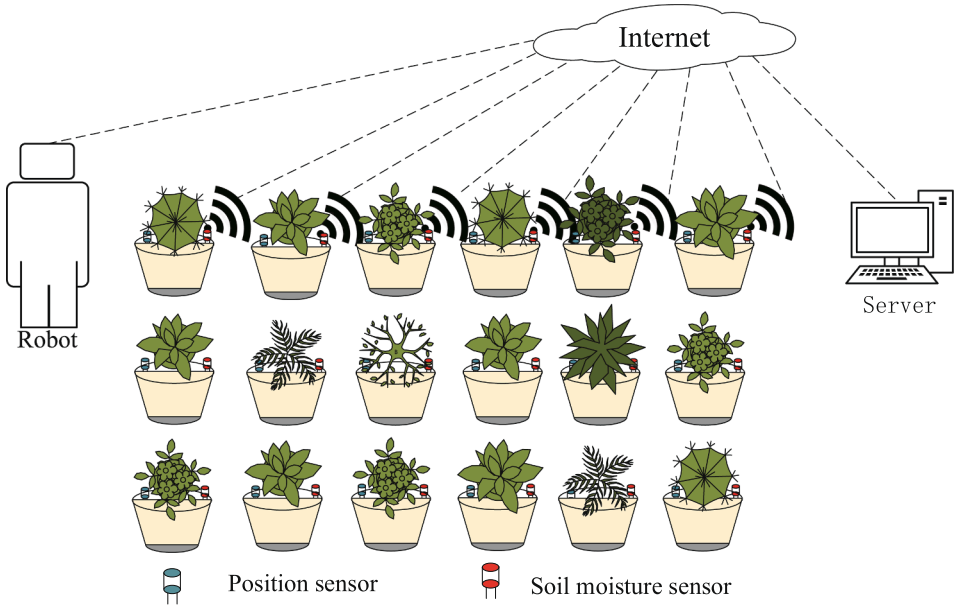

Fig. 1. Illustration of automatic watering system.The blue symbol on the left of each flowerpot indicates the position sensors, and the red symbol is the soil moisture sensors. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

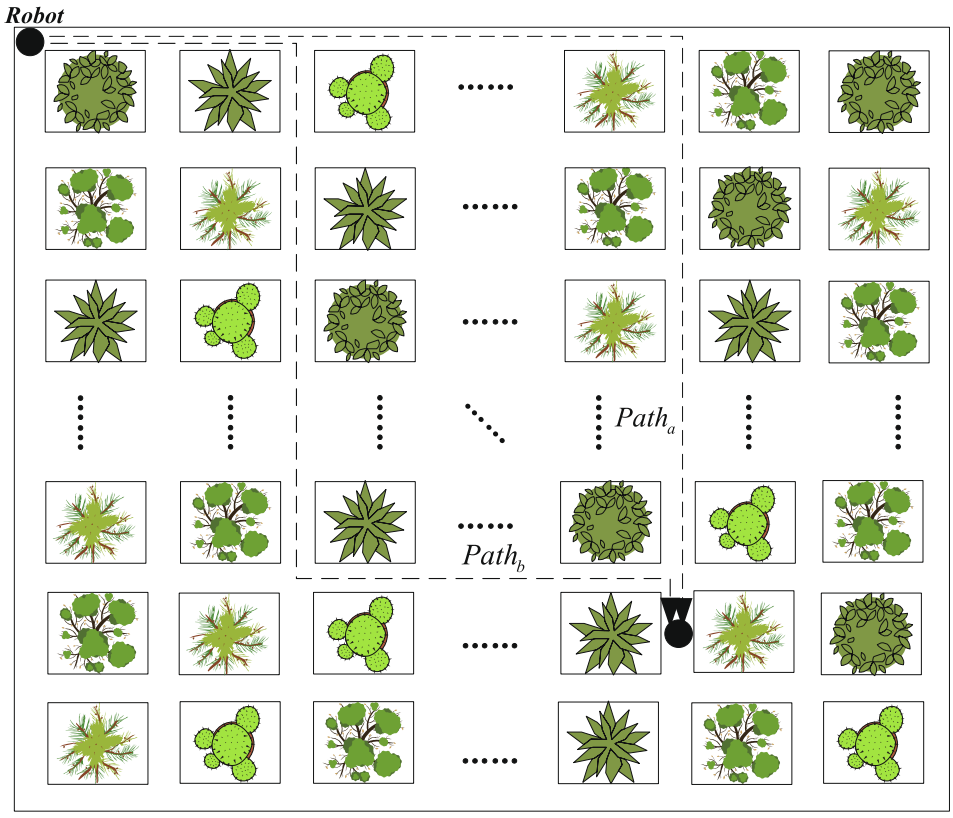

Fig. 2. Illustration of the basic watering model. $Path_a$ and $Path_b$ are two different watering paths. The later one has more corners that the former one.

total watering time for lower energy consumption (Yousefi, Hajizadeh, & Norbakhsh Soltani, 2019), the robot is required to plan watering path carefully. To represent the garden with abstract symbols, Fig. 2 can be abstracted as a position matrix (1) with each symbol corresponding to one plant. Meanwhile, corresponding watering frequency matrix, interval time matrix, watering duration matrix are represented with Eq. ()()()(2)–(4), respectively.

$$P=\begin{bmatrix} P_1 & P_2 & P_3 & \dots & P_4 & P_5 & P_1 \\ P_5 & P_4 & P_2 & \dots & P_5 & P_1 & P_4 \\ P_2 & P_3 & P_1 & \dots & P_1 & P_2 & P_5 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ P_4 & P_5 & P_2 & \dots & P_1 & P_3 & P_5 \\ P_5 & P_4 & P_3 & \dots & P_2 & P_4 & P_1 \\ P_4 & P_3 & P_5 & \dots & P_2 & P_5 & P_3 \end{bmatrix} \tag{1}$$

$$C=\begin{bmatrix} C_1 & C_2 & C_3 & \dots & C_4 & C_5 & C_1 \\ C_5 & C_4 & C_2 & \dots & C_5 & C_1 & C_4 \\ C_2 & C_3 & C_1 & \dots & C_1 & C_2 & C_5 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ C_4 & C_5 & C_2 & \dots & C_1 & C_3 & C_5 \\ C_5 & C_4 & C_3 & \dots & C_2 & C_4 & C_1 \\ C_4 & C_3 & C_5 & \dots & C_2 & C_5 & C_3 \end{bmatrix} \tag{2}$$

$$I=\begin{bmatrix} I_1 & I_2 & I_3 & \dots & I_4 & I_5 & I_1 \\ I_5 & I_4 & I_2 & \dots & I_5 & I_1 & I_4 \\ I_2 & I_3 & I_1 & \dots & I_1 & I_2 & I_5 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ I_4 & I_5 & I_2 & \dots & I_1 & I_3 & I_5 \\ I_5 & I_4 & I_3 & \dots & I_2 & I_4 & I_1 \\ I_4 & I_3 & I_5 & \dots & I_2 & I_5 & I_3 \end{bmatrix} \tag{3}$$

$$W=\begin{bmatrix} W_1 & W_2 & W_3 & \dots & W_4 & W_5 & W_1 \\ W_5 & W_4 & W_2 & \dots & W_5 & W_1 & W_4 \\ W_2 & W_3 & W_1 & \dots & W_1 & W_2 & W_5 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ W_4 & W_5 & W_2 & \dots & W_1 & W_3 & W_5 \\ W_5 & W_4 & W_3 & \dots & W_2 & W_4 & W_1 \\ W_4 & W_3 & W_5 & \dots & W_2 & W_5 & W_3 \end{bmatrix} \tag{4}$$

As exhibited in Fig. 2, plant $P^{(i,j)}(i=1,2,3,\dots,N;j=1,2,3,\dots,N)$ could be a broadleaf plant, big leaf plant, medium & small leaf plant, succulent plant or miniature tree. For plant $P^{(i,j)}$, it needs to irrigate $C^{(i,j)}$ times. In addition, corresponding watering duration and interval are $W^{(i,j)}$ and $I^{(i,j)}$, respectively. That to say, the watering process of plant $P^{(i,j)}$ may last for $W^{(i,j)}$ seconds, and at least after $I^{(i,j)}$ seconds, plant $P^{(i,j)}$ can be re-watered. If the requirement is not satisfied, plant $P^{(i,j)}$ is likely to die or be rot. For watering robot, the moving time per gird length is T_m seconds. Note that considering the difficulty of moving around at the corners, as displayed in Fig. 2, $Path_a$ is preferred to $Path_b$ because the former one has less corners than the later one.

For comparison purposes, in later experimental section, Eq. (5) is defined for calculating the total watering time without optimization:

$$\begin{aligned} T_{before}=\sum_{i=1}^{N}\sum_{j=1}^{N}(C^{(i,j)}\times W^{(i,j)}+(C^{(i,j)}-1)\times I^{(i,j)} \\ +T_m\times L^{(i,j)\to(i,j+1)\|(i+1,1)})+T_m\times L^{(N,N)\to(1,1)} \end{aligned} \tag{5}$$

where $L^{(i,j)\to(i,j+1)\|(i+1,j)}$ is the path length from $P^{(i,j)}$ to $P^{(i,j+1)}$ or from $P^{(i,j)}$ to $P^{(i+1,j)}$.

To simultaneously meet various watering requirements and minimize the total watering time, Eq. (8) is introduced to describe the watering behavior. Before that, for easy explanations of Eq. (8), this paper exhibits an optimized watering path as presented with Eq. (6), which will be explained in detail in Section 4.

$$S = [P^{(1,1)}, P^{(1,2)}, P^{(1,1)}, P^{(1,3)}, \ldots\ldots, P^{(n,n-1)}, P^{(n,n)}, P^{(n,n-2)}, P^{(n,n-1)}] \quad (6)$$

where $P^{(1,1)}$ appears twice, indicating corresponding plant requires to be watered twice. The dimension of S is equal to $D = \sum_{i=1}^{N}\sum_{j=1}^{N} C^{(i,j)}$.

For plant$P^{(1,1)}$, the actual interval time is computed with Eq. (7):

$$I_{actual}^{P^{(1,1)}} = T_m \times L^{P^{(1,1)} \to P^{(1,2)}} + W^{P^{(1,2)}} + T_m \times L^{P^{(1,2)} \to P^{(1,1)}} \quad (7)$$

where $L^{P^{(1,1)} \to P^{(1,2)}} = |1-1| + |1-2|$ is the manhattan distance from $P^{(1,1)}$ to $P^{(1,2)}$. T_m is the moving time per step length. The same rules above can also go to other symbols. Detailed explanation of S can be found in Section 4.1.

The watering time after optimization is defined as follows:

$$T_{after} = \sum_{d=1}^{D}(W^{S^d} + T_m \times L^{S^d \to S^{d+1}}) + T_m \times L^{S^D \to (1,1)} \quad (8)$$

$$s.t.\ I_{actual}^{P^{(i,j)}} \geqslant I^{P^{(i,j)}} (1 \leqslant i \leqslant N, 1 \leqslant j \leqslant N)$$

It should be specially mentioned that $I_{actual}^{P^{(i,j)}} \geqslant I^{P^{(i,j)}} (1 \leqslant i \leqslant N, 1 \leqslant j \leqslant N)$ should be satisfied. It is easy to understand that if $I_{actual}^{P^{(i,j)}} < I^{P^{(i,j)}}$, for watering robot, there is no need to move away to irrigate other plants because more time will be wasted on moving and waiting. On the contrary, $I_{actual}^{P^{(i,j)}} \geqslant I^{P^{(i,j)}}$ means that the watering robot can irrigate other plants to save unnecessary waiting time.

The basic watering model above is efficient for small garden, but its adaptability to larger garden is poor. The reason is easy to understand. Image that if there exists a garden of 10 × 10 plants, the number of variables, $D = \sum_{i=1}^{N=10}\sum_{j=1}^{N=10} C^{(i,j)}$, may exceed 100. In this case, for any state-of-art optimization algorithms, they are faced with the curse of dimensionality, which is a common term in large scale optimization problems (Mohapatra, Das, & Roy, 2017; Yang et al., 2016). The curse of dimensionality may results in slow convergence or unacceptable solutions with low quality. The phenomenon of curse of dimensionality can be illustrated with the following analysis.

Assume the watering robot is irrigating plant $P^{(i,j)}$ for the first time, and it returns to plant $P^{(i,j)}$ after watering plant$P^{(i',j')}$. The actual interval time $I_{actual}^{P^{(i,j)}}$ between two watering is calculated as follows:

$$I_{actual}^{P^{(i,j)}} = T_m \times L^{P^{(i,j)} \to P^{(i',j')}} + W^{P^{(i',j')}} + T_m \times L^{P^{(i',j')} \to P^{(i,j)}} \quad (9)$$

where if $P^{(i,j)}$ is close to $P^{(i',j')}$, indicating $T_m \times L^{P^{(i,j)} \to P^{(i',j')}} + T_m \times L^{P^{(i',j')} \to P^{(i,j)}}$ is a small value, $W^{P^{(i',j')}}$ accounts for a large percentage of the actual time $I_{actual}^{P^{(i,j)}}$. However, if $P^{(i,j)}$ is far away from $P^{(i',j')}$, it can be inferred that the proportion of $(T_m \times L^{P^{(i,j)} \to P^{(i',j')}} + T_m \times L^{P^{(i',j')} \to P^{(i,j)}})$ in $I_{actual}^{P^{(i,j)}}$ will increase. Based on the analysis above, it can be known that effective watering time (excluding the moving time) is greatly influenced by the watering path. This issue is more obvious when the garden size becomes larger, which may result in the slow convergence of selected algorithm. In addition, it can be inferred that the performance of basic watering model may become worse and worse with the increasing garden size, which can be verified in subSection 5.1.

3.2. Extended watering model

To avoid the issue above, this paper attempts to incorporate a sliding window strategy into the watering model above, forming an extended watering model. The main idea of extend watering model is described as follows.

Take Fig. 2 as an example, and the modified figure is presented in Fig. 3. Firstly, the sliding window size should be determined a prior. In principle, the best sliding window size should maximize the watering efficiency as much as possible. Therefore, Eq. (8) can be used to

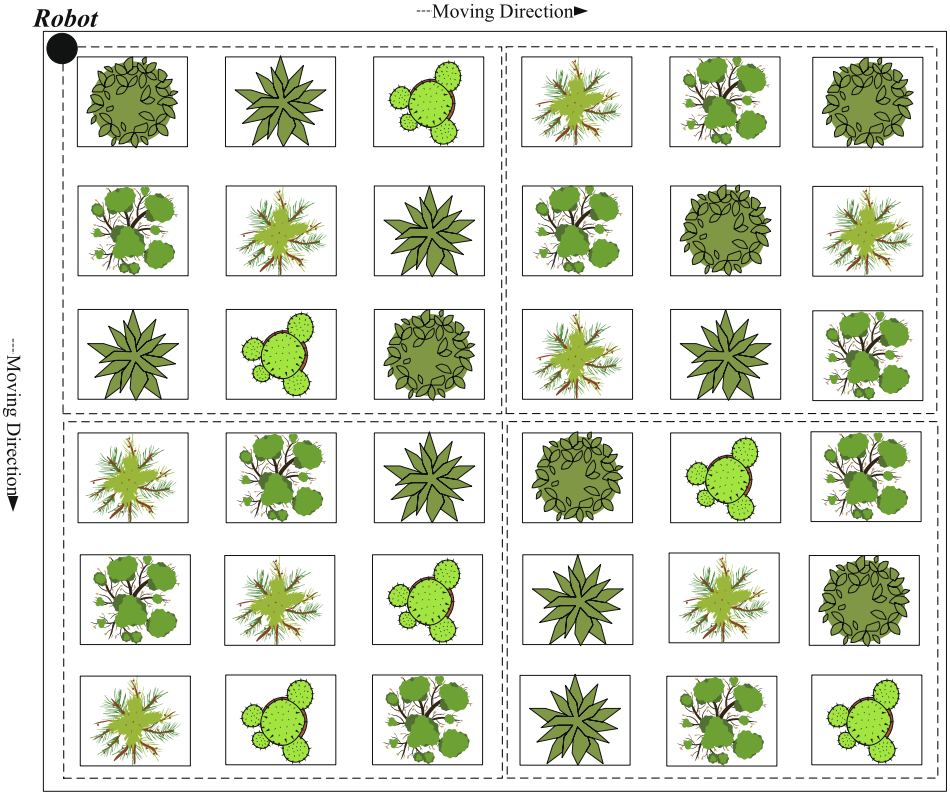

Fig. 3. Illustration of moving directions for the sliding window strategy. The sliding window firstly moves to the right of the garden at a predetermined stepsize until the boundary, and moves down to the next row until the watering process is completed.

determine the best setting of sliding window size by varying the garden size. Corresponding experiments and analysis are presented in experimental subSection 5.1. After that, a sliding window (corresponding to the dotted rectangle) is initialized at the top-right corner. As Fig. 3 exhibits, plants in the top-right window are firstly watered according to their watering requirements.After that the sliding window slides from left to right, from top to bottom with predefined step size until the sliding window reaches the left-bottom region. Note that, to avoid that the same plant is included into two neighboring sliding windows, the step size for the sliding window is generally equal to the window size. Finally, the robot is required to return to starting point. Total watering time can be computed by adding up the respective watering time of each sliding window.

Applying the sliding window strategy to position matrix, watering frequency matrix, interval time matrix, and watering duration matrix, resultant matrixes can be expressed as follows:

$$P_{(m,n)} = P^{(m:m+stepsize-1,n:n+stepsize-1)} \tag{10}$$

$$C_{(m,n)} = C^{(m:m+stepsize-1,n:n+stepsize-1)} \tag{11}$$

$$I_{(m,n)} = I^{(m:m+stepsize-1,n:n+stepsize-1)} \tag{12}$$

$$W_{(m,n)} = W^{(m:m+stepsize-1,n:n+stepsize-1)} \tag{13}$$

where $P_{(m,n)}$ $(1 \leqslant m, n \leqslant N - stepsize + 1)$ indicates corresponding sliding window, (m, n) means the top-right plant of the sliding window $P_{(m,n)}$. $P^{(m:m+stepsize-1,n+stepsize-1)}$ indicates the plants of P from row m to $m + stepsize - 1$, and from column n to $n + stepsize - 1$. $stepsize$ indicates the step size of the sliding window, as well as the sliding window size.

For sliding window $P_{(m,n)}$, the optimized watering path can be expressed as follows:

$$S_{(m,n)} = [P^{(1,1)}_{(m,n)}, P^{(1,2)}_{(m,n)}, P^{(1,1)}_{(m,n)}, P^{(1,3)}_{(m,n)}, \ldots\ldots, P^{(n,n-2)}_{(m,n)}, P^{(n,n-1)}_{(m,n)}] \tag{14}$$

According to the analysis above, the basic watering model can be improved as follows:

$$\begin{aligned} T_{after} = & \sum_{m=1}^{k}\sum_{n=1}^{k}\Big(\sum_{d=1}^{D_{(m,n)}}\big(W^{S^d_{(m,n)}}_{(m,n)} + T_m \times L^{S^d_{(m,n)} \to S^{d+1}_{(m,n)}}\big) \\ & + T_m \times L^{S^{D_{(m,n)}}_{(m,n)} \to S^{(1,1)}_{(m,n+1)||(m+1,n)}}\Big) + T_m \times L^{S^{D_{(k,k)}}_{(k,k)} \to S^{(1,1)}_{(1,1)}} \\ & s.t.\ I^{P^{(i,j)}_{(m,n)}}_{actual} \geqslant I^{P^{(i,j)}_{(m,n)}}_{(m,n)} \\ & (m \leqslant i \leqslant m + stepsize - 1, n \leqslant j \leqslant n + stepsize - 1) \end{aligned} \tag{15}$$

where $k = N/stepsize, D_{(m,n)} = \sum_{i=m}^{m+stepsize-1}\sum_{j=n}^{n+stepsize-1} C^{(i,j)}_{(m,n)}$ is the length of optimized path $S_{(m,n)}$ for current sliding window $P_{(m,n)}$.$S^d_{(m,n)}$ is the d-th dimension of $S_{(m,n)}$ for the current sliding window. $L^{S^d_{(m,n)} \to S^{d+1}_{(m,n)}}$ is the manhattan distance from $S^d_{(m,n)}$ to $S^{d+1}_{(m,n)}$. $L^{S^{D_{(m,n)}}_{(m,n)} \to S^{(1,1)}_{(m,n+1)||(m+1,n)}}$ is the distance from the last plant $S^{D_{(m,n)}}_{(m,n)}$ in current sliding window to $S^{(1,1)}_{(m,n+1)||(m+1,n)}$ in next sliding window. $L^{S^{D_{(k,k)}}_{(k,k)} \to S^{(1,1)}_{(1,1)}}$ indicates the path length from the last plant in the final sliding window to the first plant of the garden. $I^{P^{(i,j)}_{(m,n)}}_{actual} \geqslant I^{P^{(i,j)}_{(m,n)}}_{(m,n)} (m \leqslant i \leqslant m + stepsize - 1, n \leqslant j \leqslant n + stepsize - 1)$ is the constraint condition, which means, for any plant $P^{(i,j)}_{(m,n)}$ in sliding window $P_{(m,n)}$, the actual interval time $I^{P^{(i,j)}_{(m,n)}}_{actual}$ should exceed $I^{P^{(i,j)}_{(m,n)}}_{(m,n)}$.

3.3. Dynamic extended watering model

In our daily life, this case may happen that although plants have been irrigated according to watering requirements, the soil moisture still can not reach the expected standard. This case frequently happens in summer because water evaporates faster in summer than in other seasons. Generally, to reach the expected soil moisture, one more watering is needed. Therefore, considering this special case, this paper further designs a dynamic watering model based on the extended watering model, which is termed as dynamic extended watering model in later experiments. The main idea can be described as follows.

Firstly, a placeholder of $stepsize \times stepsize$ slots is used to include the information that whether corresponding plant needs to be re-watered. In terms of how to obtain the information, it can be explained as follows. As introduced above, the soil moisture sensor periodically measures the soil moisture information and sends it to the server. After the analysis by data analysis module, the server sends watering information over the Internet to the watering robot. The watering information is stored into the corresponding slot. Repeat the progress above until all the plants have send their re-watering information. It should be noted that although each plant corresponds to a slot, it does not mean that each plant has the re-watering need.

As for how to arrange the slot of each plant in the placeholder, the following description exhibits a feasible solution. To clearly describe the placeholder strategy, let $stepsize = 2$, which is determined by experimental analysis in subSection 5.1. Take a sliding window $P_{(m,n)}$ of $stepsize \times stepsize = 4$ plants as an example:

$$P_{(m,n)} = \begin{bmatrix} P_1 & P_2 \\ P_4 & P_3 \end{bmatrix} \tag{16}$$

For Eq. (14), let the last plant $P^{(n,n-1)}_{(m,n)} = P_2$. If each kind of plant has a re-watering probability $P_{watering}$, which is an empirical value and can be determined according to the practical weather conditions, an empty placeholder of 4 slots can be described with Eq. (17):

$$H_{(m,n)} = [[P_3], [P_4], [P_1], [P_2]] \tag{17}$$

where $[P_2]$ is the slot for plant P_2 . Note that the slot order is determined based on two considerations. The first one is that the last plant $P^{(n,n-1)}_{(m,n)} = P_2$ should be far away from slot $[P_2]$ in Eq. (16) to avoid the interval time of plant P_2. The second one is that $[P_3], [P_4]$ and $[P_1]$ are arranged in a counterclockwise order to avoid unnecessary moving and waiting.

Based on the description above, combine the optimized Eq. (14) with Eq. (17), and the resultant watering path can be expressed as follows:

$$C_{(m,n)} = [P^{(1,1)}_{(m,n)}, P^{(1,2)}_{(m,n)}, \ldots\ldots, P^{(n,n-1)}_{(m,n)}, [P_3], [P_4], [P_1], [P_2]] \tag{18}$$

In practice, the generated watering paths are various, and the following path is a possible one:

$$C_{(m,n)} = [P^{(1,1)}_{(m,n)}, P^{(1,2)}_{(m,n)}, \ldots\ldots, P^{(n,n-1)}_{(m,n)}, P_2] \tag{19}$$

where P_2 means that it has a re-watering need and the re-watering information is concatenated to the optimized path. However, as $P^{(n,n-1)}_{(m,n)} = P_2$, the watering robot has to wait $I^{(n,n-1)}_{(m,n)}$ according to watering requirements.

According to the description above, the total watering time can be formally expressed as follows:

$$\begin{aligned} T_{after} = & \sum_{m=1}^{k}\sum_{n=1}^{k}\Big(\sum_{d=1}^{D_{(m,n)}}\big(W^{S^d_{(m,n)}}_{(m,n)} + T_m \times L^{S^d_{(m,n)} \to S^{d+1}_{(m,n)}}\big) \\ & + \sum_{g=1}^{D_{(m,n)}}\big(W^{H^g_{(m,n)}}_{(m,n)} + T_m \times L^{H^g_{(m,n)} \to H^{g+1}_{(m,n)}} + T^{H^g_{(m,n)}}_{wait}\big) \\ & + T_m \times L^{H^{D_{(m,n)}}_{(m,n)} \to S^{(1,1)}_{(m,n+1)||(m+1,n)}}\Big) + T_m \times L^{S^{D_{(k,k)}}_{(k,k)} \to S^{(1,1)}_{(1,1)}} \end{aligned} \tag{20}$$

$$s.t. I^{S^d_{(m,n)}}_{actual} \geqslant I^{S^d_{(m,n)}}_{(m,n)}$$

$$(1 \leqslant m \leqslant \frac{N}{stepsize}, 1 \leqslant n \leqslant \frac{N}{stepsize}, 1 \leqslant d \leqslant D_{(m,n)})$$

where $k = N/stepsize, D_{(m,n)}$ is the actual dimension of the placeholder. $W^{H^g_{(m,n)}}_{(m,n)}$ indicates the watering duration of plant $H^g_{(m,n)}(1 \leqslant g \leqslant D_{(m,n)})$, $L^{H^{D_{(m,n)}}_{(m,n)} \to S^{(1,1)}_{(m,n+1)||(m+1,n)}}$ means the path length from the last plant $H^{D_{(m,n)}}_{(m,n)}$ of placeholder $H_{(m,n)}$ to the first plant $S^{(1,1)}_{(m,n+1)||(m+1,n)}$ of the next sliding window. $T^{H^g_{(m,n)}}_{wait}$ is the actual waiting time, which can be defined as follows:

$$T^{H^g_{(m,n)}}_{wait} = \begin{cases} 0, I^{H^g_{(m,n)}}_{actual,(m,n)} \geqslant I^{H^g_{(m,n)}}_{(m,n)} \\ I^{H^g_{(m,n)}}_{(m,n)} - I^{H^g_{(m,n)}}_{actual,(m,n)}, otherwise \end{cases} \tag{21}$$

Eq. (21) indicates that if the actual interval time of plant $H^g_{(m,n)}$ is less than the defined interval time, robot has to wait $I^{H^g_{(m,n)}}_{(m,n)} - I^{H^g_{(m,n)}}_{actual,(m,n)}$; otherwise, the watering robot can directly irrigate plant $H^g_{(m,n)}$. Meanwhile, it can be concluded that due to the existence of possible waiting time $I^{H^g_{(m,n)}}_{(m,n)} - I^{H^g_{(m,n)}}_{actual,(m,n)}$, the average time saving rate may decrease unexpectively, which can be verified by later experiments and analyses.

4. Genetic algorithm with neighbor exchanging strategy for watering models

Genetic Algorithm(*GA*) (Zhang et al., 2020) is originally proposed for single objective optimization problems (Zhang et al., 2021). Since its publication, *GA* has gained a lot of recognitions due to its excellent performance on benchmark problems. In many applications, *GA* has also exhibited outstanding performance compared to other optimization methods. As shown in Fig. 4, *GA* involves various components, such as initialization of population, selection operator, crossover operator, mutation operator and population updating. In *GA*, there are multiple coding methods for individual representation, such as binary coding and integer coding. Due to the characteristics of watering models, this paper employs integer coding. In the following subsections, the main components of *GA* will be redesigned one by one.

4.1. Individual representation and population initialization

For simplicity, given four kinds of plants, corresponding sliding window and watering frequency matrix, watering duration matrix and interval time matrix are presented as follows:

$$P_{(m,n)} = \begin{bmatrix} P_1 & P_2 \\ P_4 & P_3 \end{bmatrix} \tag{22}$$

$$C_{(m,n)} = \begin{bmatrix} 3 & 3 \\ 2 & 2 \end{bmatrix} \tag{23}$$

$$W_{(m,n)} = \begin{bmatrix} W_1 & W_2 \\ W_4 & W_3 \end{bmatrix} \tag{24}$$

$$I_{(m,n)} = \begin{bmatrix} I_1 & I_2 \\ I_4 & I_3 \end{bmatrix} \tag{25}$$

Based on the provided data, a possible watering path of $\sum_{i=1}^{2}\sum_{j=1}^{2} C^{(i,j)}_{(m,n)}$ variables is as follows:

$$[P^{(1,1)}_{(m,n)}, P^{(1,2)}_{(m,n)}, P^{(1,1)}_{(m,n)}, P^{(1,1)}_{(m,n)}, P^{(1,2)}_{(m,n)}, P^{(1,3)}_{(m,n)}, P^{(1,2)}_{(m,n)}, P^{(1,4)}_{(m,n)}, P^{(1,3)}_{(m,n)}] \tag{26}$$

The watering path above means that the watering robot needs to irrigate these plants sequentially. The frequency of occurrence of plant $P^{(i,j)}_{(m,n)}$ is determined by corresponding $C^{(i,j)}_{(m,n)}$. For plant $P^{(1,1)}_{(m,n)}$, the actual interval time of the first two watering is calculated with the following equation:

$$I^{P^{(1,1)}_{(m,n)}}_{actual} = T_m \times L^{P^{(1,1)}_{(m,n)} \to P^{(1,2)}_{(m,n)}} + W^{P^{(1,2)}_{(m,n)}}_{(m,n)} + T_m \times L^{P^{(1,2)}_{(m,n)} \to P^{(1,1)}_{(m,n)}} \tag{27}$$

According to constraint conditions, $I^{P^{(1,1)}_{(m,n)}}_{actual}$ should exceed $I^{(1,1)}_{(m,n)}$. It is obvious that the actual interval time between the second and third watering of $P^{(1,1)}_{(m,n)}$ is zero, which is unable to satisfy the watering requirements. Luckily, the generated watering path is an initialized sequence, and it can be further optimized using the method introduced as folows.

Population initialization can be described as follows:

(1) Determine the population size;
(2) Randomly generate a path sequence, where plant $P^{(i,j)}_{(m,n)}$ $(i = 1, 2; j = 1, 2)$ should appear $C^{(i,j)}_{(m,n)}$ time(s) because watering frequency of $P^{(i,j)}_{(m,n)}$ is $C^{(i,j)}_{(m,n)}$ according to watering requirements;
(3) Check whether the generated path sequence is acceptable according to the interval time constraint;
(4) If acceptable, the generated path sequence is included into the population; Otherwise, return to step (2);
(5) If the number of acceptable individuals reaches the predefined population size, output the population; Otherwise, return to step (2).

It should be noted that the generated path is just an acceptable one, and corresponding total watering time may be unsatisfying. Therefore, it should be optimized accordingly.

4.2. Neighbor exchanging strategy for crossover operator

To have an effective crossover operator, this paper proposes to employ asexual crossover operator. Firstly, randomly generate two position indexes N_1 and N_2, satisfying $1 \leqslant N_1 < N_2 \leqslant \sum_{i=1}^{2}\sum_{j=1}^{2} C^{(i,j)}_{(m,n)} = 10$, where $\sum_{i=1}^{2}\sum_{j=1}^{2} C^{(i,j)}_{(m,n)}$ is the watering path length. Let $N_1 = 4, N_2 = 9$, and subsequence(28) is picked up from (26).

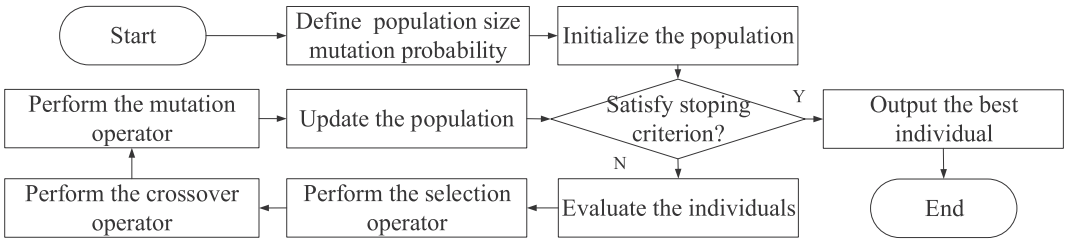

Fig. 4. Flow chart of genetic algorithm.

$$[P_{(m,n)}^{(1,1)}, P_{(m,n)}^{(1,2)}, P_{(m,n)}^{(1,3)}, P_{(m,n)}^{(1,2)}, P_{(m,n)}^{(1,4)}, P_{(m,n)}^{(1,3)}] \tag{28}$$

After that, neighboring plants swap positions with each other. An illustrative example is presented as follows:

$$[P_{(m,n)}^{(1,1)} \rightleftarrows P_{(m,n)}^{(1,2)}, \quad P_{(m,n)}^{(1,3)} \rightleftarrows P_{(m,n)}^{(1,2)}, \quad P_{(m,n)}^{(1,4)} \rightleftarrows P_{(m,n)}^{(1,3)}] \tag{29}$$

where $\rightleftarrows$ indicates the operation of swapping the neighboring plants. In the subsequence above, the number of plants is even. However, the last plant should remain unchanged if the number is odd. The resultant subsequence is as follows:

$$[P_{(m,n)}^{(1,2)}, P_{(m,n)}^{(1,1)}, P_{(m,n)}^{(1,2)}, P_{(m,n)}^{(1,3)}, P_{(m,n)}^{(1,3)}, P_{(m,n)}^{(1,4)}] \tag{30}$$

Note that the generated sequence may not meet the interval constraints. In this case, N_1 and N_2 should be regenerated until the generated individual meets the corresponding constraints. In later experiment section, genetic algorithm with neighbor exchanging strategy is termed as *NEGA*.

4.3. Mutation operator

Standard mutation operator in *GA* is firstly to determine the mutation chromosome, then replace it with new generated chromosome. Obviously, it is not applicable to this coding method. Thus, this paper proposes to randomly generate two chromosome positions, then swap the two chromosomes. For example, assuming $M_1 = 3, M_2 = 8$ are randomly generated chromosome positions, the mutation process can be described as follows:

$$[P_{(m,n)}^{(1,1)}, P_{(m,n)}^{(1,2)}, \left\|P_{(m,n)}^{(1,1)}\right\|, P_{(m,n)}^{(1,1)}, P_{(m,n)}^{(1,2)}, P_{(m,n)}^{(1,3)}, P_{(m,n)}^{(1,2)}, \left\|P_{(m,n)}^{(1,4)}\right\|, P_{(m,n)}^{(1,3)}, P_{(m,n)}^{(1,4)}] \tag{31}$$

where $\|\|$ is used to highlight the selected plants. Note that if the generated sequence is unable to satisfy the constraint conditions, M_1 and M_2 should be regenerated accordingly until the generated sequence meets constraints. The mutation probability P_m in *GA* is used to determine whether one individual is selected to conduct mutation operation. The newly generated watering path can be expressed as follows:

$$[P_{(m,n)}^{(1,1)}, P_{(m,n)}^{(1,2)}, P_{(m,n)}^{(1,4)}, P_{(m,n)}^{(1,1)}, P_{(m,n)}^{(1,2)}, P_{(m,n)}^{(1,3)}, P_{(m,n)}^{(1,2)}, P_{(m,n)}^{(1,1)}, P_{(m,n)}^{(1,3)}, P_{(m,n)}^{(1,4)}] \tag{32}$$

4.4. Population update

Preserving high-quality individuals are of great necessity. It can not only prevent the loss of elite individuals, but also ensure fast convergence. The detailed process can be described as follows:

Assume the population size is N. First, combine the generated offspring and corresponding parent populations into one population. Thus, the combined population has $2 \times N$ individuals. Following that, calculate the fitness value of each individual. Sort the individuals in ascending order. Select the first N individuals as the next offspring population.

Tournament selection strategy (Kumar et al., 2019) is a regular strategy, which is commonly used for the reproduction of new individuals. In this paper, this strategy is not intended to be specially designed. Therefore, the details of tournament selection strategy are no longer introduced in this paper. Readers are encouraged to refer to original paper (Kumar et al., 2019). In terms of the time complexity, it can be analyzed as follows. For a population of N individuals, the evaluation cost is $O(ND)$, where $D = \sum_{i=1}^{stepsize}\sum_{j=1}^{stepsize} C_{(m,n)}^{(i,j)}$ is the dimension of each individual. The sorting for selection operator and population update needs a runtime of $O(N^2)$. In the worst case, the crossover operator and mutation operator need a runtime of $O(ND)$. Therefore, the total time complexity of *NEGA* is $O(N^2)$.

5. Experiments and analysis

In this section, three experiments will be conducted. The first one is to analyze the effectiveness and drawback of basic watering model, as well as to determine the best setting of sliding window size. After that, the second experiment is presented to illustrate the performance of extended watering model, as well as to verify the effectiveness of sliding window strategy. Following that, experiments on dynamic extended watering model are conducted to verify the performance of this model in various weather conditions. Before experiments, related parameters and experimental environment are introduced, respectively.

For *NEGA*, the mutation probability P_m is set to 0.1. Each algorithm is run 20 times. In subSection 5.1, maximum generation 500 is used as stopping criteria. The population size is set to 50. Table 1 presents different kinds of plants used in experiments. Corresponding duration, interval time, and watering frequency are presented, respectively. Note that the soil used in the experiments is clay. The moving time T_m for each grid length is set to one second. Since the experiments involve servers, robot, various plants and a lot of sensors, it is impossible to verify these models in real scenarios, especially for the dynamic extended model which is closely related to the weather conditions. To tackle this problem, simulation environment is set up on computer with Matlab R2018a, Windows 10 system and Intel Core i5-3210 M processor.

In order to have a uniform metric, this paper defines a time saving rate as an indicator, and it can be mathematically described as follows:

$$R = \frac{T_{before} - T_{after}}{T_{before}} \times 100\% \tag{33}$$

where T_{before} is the total watering time before optimization. To be specific, the watering robot firstly waters the first plant, and then waits a few seconds for the second watering until the watering requirements are satisfied. Repeat the watering process above until all the plants are watered. T_{after} indicates the optimized total watering time. In general, $T_{before} > T_{after}$, which indicates the proposed model is effective in optimizing the watering path. On the contrary, $T_{before} \leqslant T_{after}$, namely $R \leqslant 0$, indicating the optimized watering path performs worse than randomly generated path.

5.1. Analysis of basic watering model

In this subsection, various garden sizes are tested using the basic watering model, and they are set to $2 \times 2, 3 \times 3, 4 \times 4, 5 \times 5, 6 \times 6, 7 \times 7, 8 \times 8, 9 \times 9$ and 10×10, respectively. All plants belong to the five categories listed in Table 1. In terms of plant locations, they are randomly initialized. In reality, the locations of plants are determined by sensors. However, in these experiments, the above information is known a prior.

Fig. 5 presents the averages of time saving rates over 20 runs. x-axis indicates the garden size, while y-axis is the average time saving rate. From Fig. 5, it can be known that the basic watering model is effective for garden with size less than 9×9, but the time saving rate linearly decrease with the garden size increasing. The phenomenon is consistent with the analysis in subSection 3.2. It is obviously ineffective for garden with size more than 10×10 because the time saving rate is negative. In

Table 1
Illustration of experimental plants.

Plant category	Watering duration (s)	Interval time (s)	Watering frequency (Times)
Broadleaf plants	8	10	3
Big leaf plants	8	9	3
Medium &small leaf plants	7	10	3
Succulent plants	3	0	1
Miniature trees	4	1	2

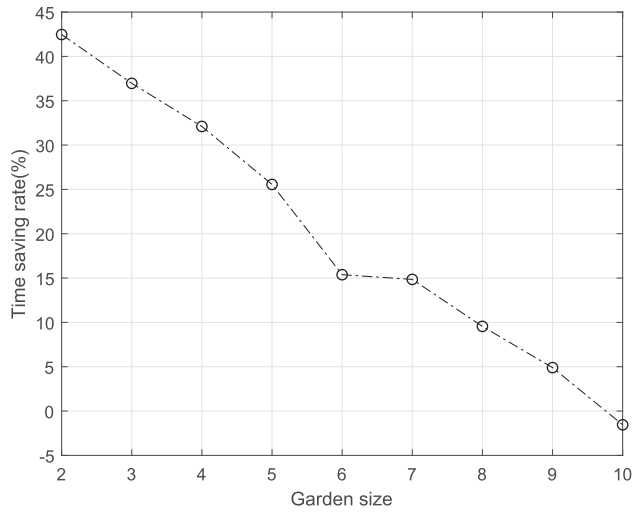

Fig. 5. Comparison of time saving rates. Garden size varys from 2×2 to 10×10.

other words, the optimized watering path performs worse than the randomly generated one. As discussed above, the reason for the poor adaptability is due to the curse of dimensionality.

However, an interesting fact is observed from Fig. 5. Garden size $= 2 \times 2$ has the highest time saving rate 42.4%, which inspires us to employ 2×2 as the sliding window size in the later experiments. Further, to show how the generation affects the basic watering model, Fig. 6 presents the dynamic convergence curve. From Fig. 6, it can be observed that the time saving rate is able to reach the highest value at generation 100. More evaluations (generations) do not contribute to the further improvement of the time saving rate. Therefore, in later experiments, the maximal generation for each sliding window is set to 100.

One may wonder that the poor adaptability is likely due to unreasonable generation setting. To further investigate the effect of generation on the basic watering model, this paper further increases generation to 1500. Corresponding experimental results are exhibited in Fig. 7. Note that y-axis indicates the average time saving rate over 20 runs and it is a negative value. As can be seen, although the time saving rate is significantly improved, the final result is still negative even at generation 1500, illustrating that the basic watering model is less effective in dealing with larger garden.

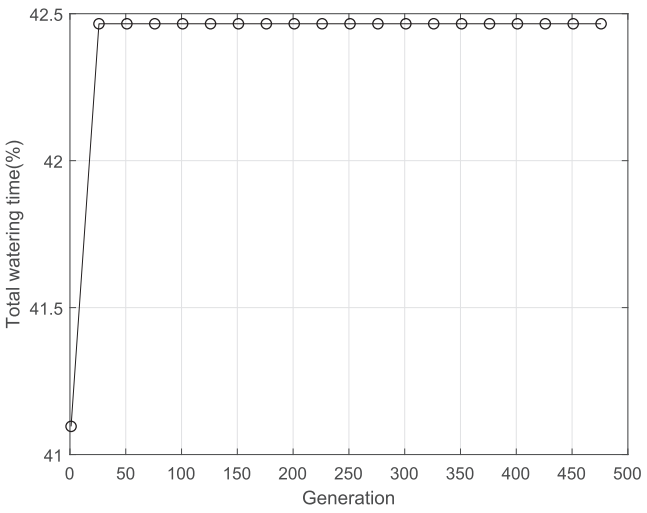

Fig. 6. Convergence curve for garden size $= 2 \times 2$. The generation is increased to 500.

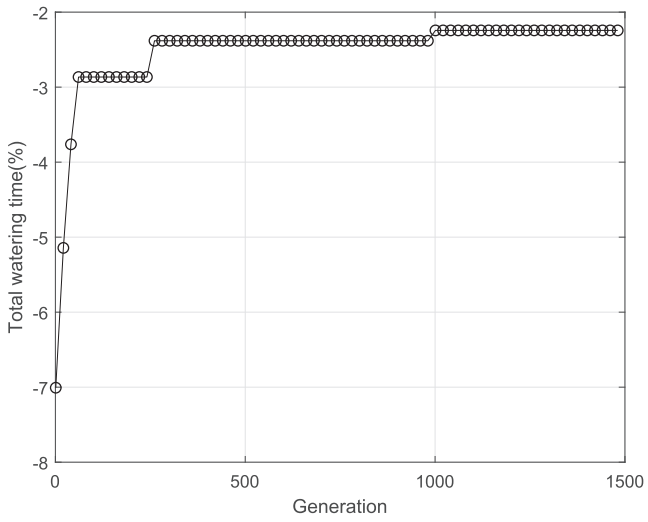

Fig. 7. Convergence curve for garden size $= 10 \times 10$. The generation is increased to 1500.

5.2. Experiment and analysis of extended watering model

This subsection is intended to fully test the extended watering model. Here, different garden sizes, $2 \times 2, 4 \times 4, 6 \times 6, 8 \times 8, 10 \times 10, 12 \times 12, 14 \times 14, 16 \times 16, 18 \times 18, 22 \times 22, 26 \times 26$ and 30×30 are discussed, respectively. As discussed in subsection above, 2×2 is relatively the best sliding window size, and 100 iterations are used as the stopping criterion. Other parameters remain the same as explained above.

In Fig. 8, x-axis means the garden size, while y-axis indicates the average time saving rate. It is notable that garden size is different from the sliding window size because they have essential difference as explained in subSection 3.2. It can be seen that, from Fig. 8, the extended watering model performs very well in all cases. The maximum time saving rate hopefully reaches 41.2%, and the minimum time saving rate is still more than 37%. In addition, the performance of extended watering model does not linearly decrease as the basic watering mode does. Moreover, even if the garden size increases to 30×30, the performance is still outstanding and does not decrease significantly, fully illustrating the effectiveness of sliding window strategy and the outstanding adaptability of extended watering model.

Fig. 9 shows the dynamic convergence curves for different garden

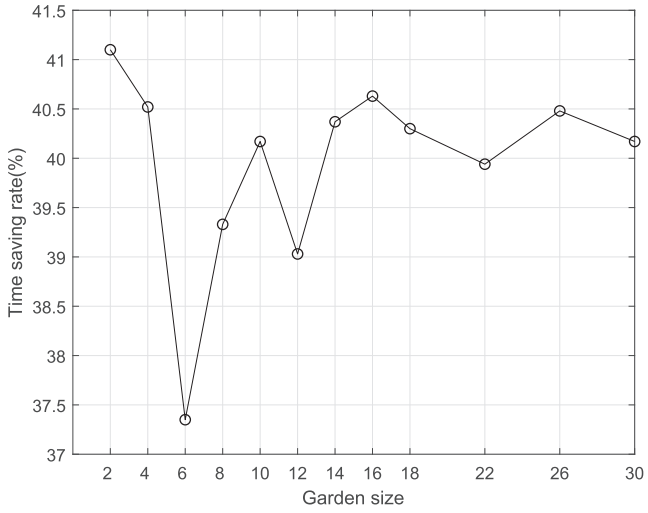

Fig. 8. Comparison of time saving rates for extended watering model. Garden size varys from 2×2 to 30×30.

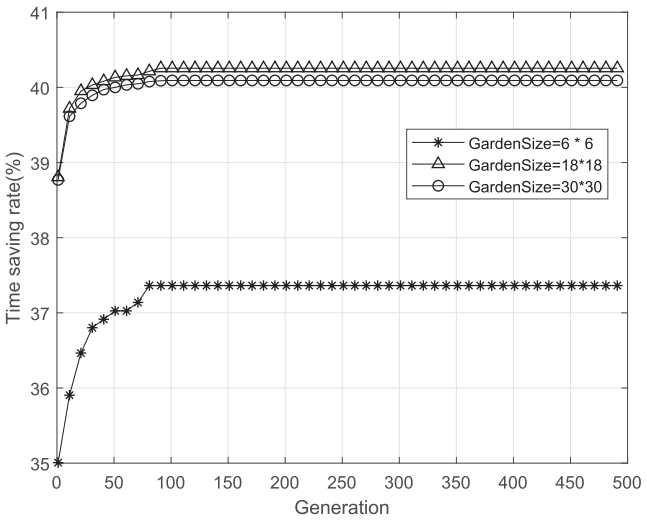

Fig. 9. Convergence curves for various garden sizes. Garden size $6 \times 6, 18 \times 18$ and 30×30 are tested, respectively.

sizes 6×6, 18×18 and 30×30. Maximum generation is set to 500. From the convergence curves, it can be known that the extended watering model is capable of converging to the best value at generation 100, and more iterations can no longer improve the final time saving rate. The high adaptability of extended watering model to garden size is verified accordingly.

5.3. Experiment and analysis of dynamic extended watering model

In this subsection, the dynamic extended watering model is systematically tested. Because the real weather conditions are difficult to simulate, this paper employs a probability to describe the re-watering conditions. For each plant, the re-watering probability (termed as $P_{watering}$) is respectively set to 0.2, 0.4, 0.6, 0.8 and 1.0 to test the robustness of the dynamic extended watering model. In addition, various garden sizes are also employed as comparisons of dynamic extended watering model in different environments. The sliding window size is set to 2×2, and the stopping criterion is set to 100. All the experimental results are visually plotted in Fig. 10.

It can be seen from Fig. 10, in various cases, the performance of dynamic extended watering model differs slightly. Relatively speaking, when $P_{watering}$=0.2, the overall performance of dynamic extended watering model is poor. The performance gradually gets improved with the increase of probability $P_{watering}$. It can also observed that the dynamic extended watering model with $P_{watering}$=1.0 reaches the highest time saving rate. However, it is hard to say that the dynamic extended watering model with $P_{watering}$=1.0 is better than that with $P_{watering}$=0.2 because these cases above belong to different weather conditions. Additionally, in terms of the changing rate of curves, dynamical extended watering model has the lowest time saving rate when the garden size is equal to 6×6 in all cases, which is consistent with the analysis in subSection 5.2.

To make it clear that how the dynamic extended watering model performs in dealing with larger gardens, this paper tests gardens size varying from 30×30 to 100×100. Corresponding experimental results are plotted in Fig. 11. From Fig. 11, it can be seen that all curves tend to be stable with no significant fluctuations, which illustrates that the performance of the dynamic extended watering model no longer gets improved in terms of the time saving rate. Therefore, it can be said that the placeholder strategy is effective, the dynamic watering model is able to deal with various cases according to the analysis above.

In summary, in terms of the garden size, it can be concluded that the extended watering model is more flexible and suitable in tackling watering problem compared with the basic watering model, and the total watering time can be greatly reduced. Considering the dynamic watering requirements, dynamic extended watering model can effectively deal with watering path optimization problem in special weather.

6. Conclusion and further work

The curse of dimensionality and insufficient irrigation are two major concerns for practical watering models. An effective model is of great importance to ensure the watering efficiency. To remedy these issues, this paper tries to propose two strategies. Firstly, to improve the adaptability of basic watering model to large gardens, the extended watering model is proposed based on the sliding window strategy. Further, to adapt to dynamic watering requirements, this paper designs a placeholder strategy and incorporates it to the extended watering model. In addition to these models, an improved genetic algorithm is proposed to tackle the proposed models. In experimental section, three experiments are conducted. According to the analysis of basic watering model, the best setting of sliding window size is determined. The second experiment systematically investigates the adaptability of the extended watering model. The experimental results reveal that the extended

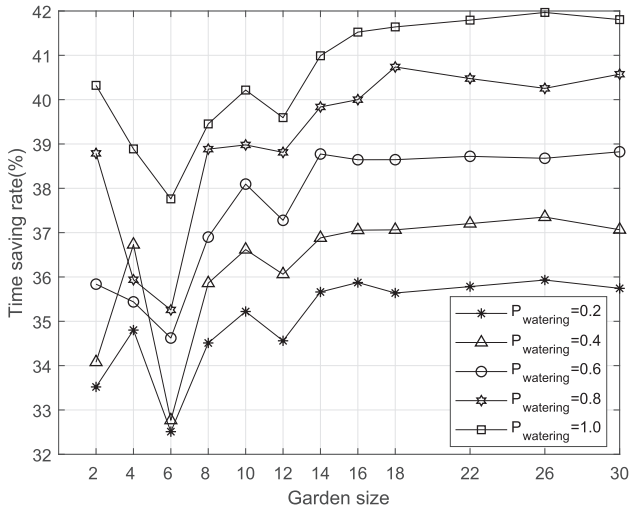

Fig. 10. Convergence curve for medium-sized gardens. Garden size varys from 2×2 to 30×30, and $P_{watering}$ varys from 0.2 to 1.0.

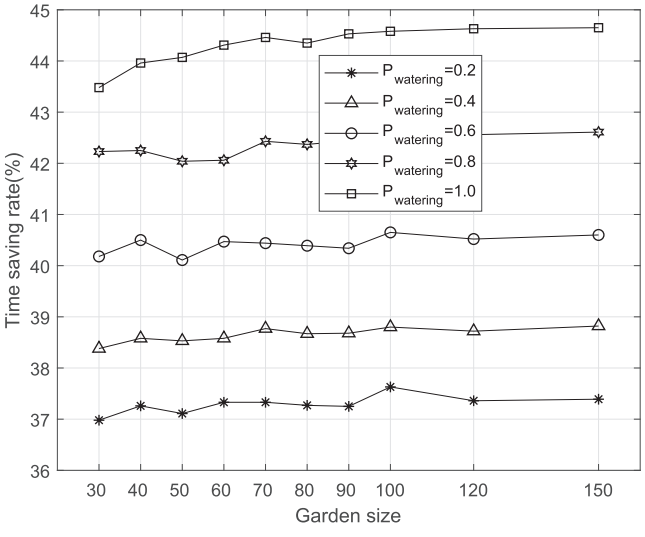

Fig. 11. Convergence curve for oversized gardens. Garden size varys from 30×30 to 150×150,and $P_{watering}$ varys from 0.2 to 1.0 with interval = 0.2.

watering model is effective in dealing with different garden sizes. The last experiment on dynamical extended watering model concludes that this model still performs very well in special weathers, and can achieve a promising time saving rate as well.

In reality, there is a problem with these models above in tackling larger gardens. As can be seen from Fig. 11, although promising time saving rate has been reached, the overall watering time is still high. That is to say, a long-time watering process is inevitable. Therefore, employing multiple robots to collaborate to complete the watering process is a practical and feasible way. However, how to construct an effective watering model to make multiple watering robots collaborate to complete the watering tasks is still a challenging task. In addition, the location method is based on the position sensors, which is a simple method but costs much money. So, these sensors can be replaced with advanced deep learning methods. Deep learning can be used to recognize the plant category and then further locate the plant. Thus, the further work is focused on multi-robot collaboration problem and plant recognition to lower time cost and material cost.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (grant number: 71771176 and 61503287), Natural Science Foundation of Shanghai, China(grant number: 19ZR1479000, 20692191200), Fundamental Research Funds for the China Central Universities(grant number: 22120190202) and Science and technology Winter Olympic project (grant number: 2018YFF0300505).

References

Abdi, F., Chen, C., Hasan, M., Liu, S., Mohan, S., & Caccamo, M. (2019). Preserving physical safety under cyber attacks. *IEEE Internet of Things Journal, 6*, 6285–6300.

Alvarez, A., Caiti, A., & Onken, R. (2004). Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering, 29*, 418–429.

Anthony, S. (1994). Optimal and efficient path planning for partially-known environments. In *Proceedings of Robotics and Automation* (pp. 203–220). IEEE.

Azhar, F. C., Irawan, B., & Saputra, R. E. (2017). Controlling and monitoring ornamental plants care remotely using android application. In *Proceedings of Wireless and Mobile (APWiMob) 2017 IEEE Asia Pacific Conference on IEEE* (pp. 12–18).

Borges, Y. G., Miyazawa, F. K., Schouery, R. C., & Xavier, E. C. (2020). Exact algorithms for class-constrained packing problems. *Computers & Industrial Engineering, 144*, 106455.

Cai, X., Geng, S., Wu, D., Cai, J., & Chen, J. (2020). A multi-cloud model based many-objective intelligent algorithm for efficient task scheduling in internet of things. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2020.3040019

Chen, M., & Lin, C. (2019). Standby power management of a smart home appliance by using energy saving system with active loading feature identification. *IEEE Transactions on Consumer Electronics, 65*, 11–17.

Cui, Z., Fei, X., Zhang, S., Cai, X., Cao, Y., Zhang, W., & Chen, J. (2020). A hybrid blockchain-based identity authentication scheme for multi-wsn. *IEEE Transactions on Services Computing, 13*, 241–251.

Cui, Z., Xu, X., Xue, F., Cai, X., Cao, Y., Zhang, W., & Chen, J. (2020). Personalized recommendation system based on collaborative filtering for iot scenarios. *IEEE Transactions on Services Computing*. https://doi.org/10.1109/JIOT.2020.3040019

Deb, K., & Tiwari, S. (2008). Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research, 185*, 1062–1087.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*, 269–271.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26*, 1–13.

Garau, B., Alvarez, A., & Oliver, G. (2005). Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an a* approach. In *Proceedings of the 2005 IEEE international conference on robotics and automation* (pp. 194–198). IEEE.

Gaw, D., & Meystel, A. (1986). Minimum-time navigation of an unmanned mobile robot in a 2–1/2d world with obstacles. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 1670–1677). IEEE.

Gerke, M. (1999). Genetic path planning for mobile robots. In *Proceeding of American Control Conference* (pp. 2424–2429).

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of things: A vision, architectural elements, and future directions. *Future Generation Computer Systems, 29*, 1645–1660.

Gunathillake, A., Huang, H., & Savkin, A. V. (2019). Sensor-network-based navigation of a mobile robot for extremum seeking using a topology map. *IEEE Transactions on Industrial Informatics, 15*, 3962–3972.

Hart, P. E., Nilsson, N. J., & Bertram, R. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics, 4*, 100–107.

Hua, J., Wang, X., Kang, M., Wang, H., & Wang, F. (2018). Blockchain based provenance for agricultural products: a distributed platform with duplicated and shared bookkeeping. In *2018 IEEE Intelligent Vehicles Symposium (IV)* (pp. 97–101). IEEE.

Jia, M., Li, D., Yin, Z., Guo, Q., & Gu, X. (2019). High spectral efficiency secure communications with non-orthogonal physical and multiple access layers. *IEEE Internet of Things Journal, 6*, 5954–5961.

Kammoun, A., Hamidouche, W., Belghith, F., Nezan, J. F., & Nouri, M. (2018). Hardware design and implementation of adaptive multiple transforms for the versatile video coding standard. *IEEE Transactions on consumer electronics, 64*, 424–432.

Kavraki, L., & Latombe, J. C. (1994). Randomized preprocessing of configuration space for fast path planning. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 2138–2145).

Kim, D., & Lee, J. (2019). Efficient and secure device clustering for networked home domains. *IEEE Transactions on consumer electronics, 65*, 224–232.

Kumar, S. R., Sudip, M., & Singh, N. R. (2019). Senspnp: Seamless integration of heterogeneous sensors with iot devices. *IEEE Transactions on Consumer Electronics, 65*, 205–214.

Mohapatra, P., Das, K. N., & Roy, S. (2017). A modified competitive swarm optimizer for large scale optimization problems. *Applied Soft Computing, 59*, 340–362.

Nuwan, G., Cheng, C., & Tse, C. K. (2015). A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains. *IEEE Transactions on Industrial Informatics, 11*, 601–611.

Pavithra, D. S., & Srinath, M. S. (2014). Gsm based automatic irrigation control system for efficient use of resources and crop planning by using an android mobile. *IOSR Journal of Mechanical and Civil Engineering, 11*, 49–55.

Petres, C., Pailhas, Y., Patron, P., Petillot, Y., Evans, J., & Lane, D. (2007). Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Transactions on Robotics, 23*, 331–341.

Pienaar, J. P., Fisher, R. M., & Hancke, G. P. (2015). Smartphone: The key to your connected smart home. In *Proceedings of IEEE 13th International Conference on Industrial Informatics (INDIN)* (pp. 999–1004). IEEE.

Plonski, P. A., Tokekar, P., & Isler, V. (2013). Energy-efficient path planning for solar-powered mobile robots. *Journal of Field Robotics, 30*, 583–601.

Ram, V. V. H., Vishal, H., Dhanalakshmi, S., & Vidya, P. M. (2015). Regulation of water in agriculture field using internet of things. In *2015 IEEE Technological Innovation in ICT for Agriculture and Rural Development (TIAR)* (pp. 112–115). IEEE.

Sharma, P., Shinde, S., & Chougle, S. (2015). Sensor based automated irrigation system with iot. *International Journal of Computer Science and Information Technologies, 6*, 5331–5333.

Shawn, A. (2014). Gardenpi: Garden care with raspberry pi. Website. https://spin.atomicobject.com/2014/06/28/raspberry-pi-gardening.

Yang, Q., Chen, W., Gu, T., Zhang, H., Deng, J. D., & Li, Y. (2016). Segment-based predominant learning swarm optimizer for large-scale optimization. *IEEE Transactions on Cybernetics, 47*, 2896–2910.

Yildiz, H. U. (2019). Maximization of underwater sensor networks lifetime via fountain codes. *IEEE Transactions on Industrial Informatics, 15*, 4602–4613.

Yousefi, M., Hajizadeh, A., & Norbakhsh Soltani, M. (2019). A comparison study on stochastic modeling methods for home energy management systems. *IEEE Transactions on Industrial Informatics, 15*, 4799–4808.

Zhang, M., Wang, L., Cui, Z., Liu, J., Du, D., & Guo, W. (2020). Fast nondominated sorting genetic algorithm ii with lévy distribution for network topology optimization. Mathematical Problems in Engineering, 2020.

Zhang, M., Wang, L., Guo, W., Li, W., Li, D., Hu, B., & Wu, Q. (2021). Many-objective evolutionary algorithm based on relative non-dominance matrix. *Information Sciences, 547*, 963–983.