Simulation and optimization of robotic tasks for UV treatment of diseases in horticulture

Merouane Mazar[1] · **M'hammed Sahnoun**[1] · **Belgacem Bettayeb**[2] · **Nathalie Klement**[3] · **Anne Louis**[1]

Abstract
Robotization is increasingly used in the agriculture since the last few decades. It is progressively replacing the human workforce that is deserting the agricultural sector, partly because of the harshness of its activities and health risks they may present. Moreover, robotization aims to improve efficiency and competitiveness of the agricultural sector. However, it leads to several research and development challenges regarding robots supervision, control and optimization. This paper presents a simulation and optimization approach for the optimization of robotized treatment tasks using type-c ultraviolet radiation in horticulture. The optimization of tasks scheduling problem is formalized and a heuristic and a genetic algorithms are proposed to solve it. These algorithms are evaluated compared to an exact method using a multi-agent-based simulation approach. The simulator takes into account the evolution of the disease during time and simulates the execution of treatment tasks by the robot.

Keywords Scheduling · Simulation · Optimization · Multi agent system · UV-c treatment

1 Introduction

Since the dawn of time, humans are trying to improve the yield of agricultural activities and to make them less painful, starting by using animals, then machines, and, nowadays, robots. Several research works deal with the design and the development of robots in the agricultural field. Farming robots can be found in many agricultural activities, from plant cultivation to harvest (Sistler 1987). The most used robots in agriculture are sprayers and combine harvesters. Several laboratories are developing methods to improve and facilitate the cultivation of plants. It is important to notice that the agriculture sector is not limited to the cultivation of fruits and vegetables, but it also includes other related activities such as food industry, spices, tissues and basic elements of drugs (Oberti et al. 2016).

Downy and powdery mildew are two types of fungi of the same family that usually contaminate plants. There are small differences between them about the manner they infect the leaves of plants. Downy mildew is characterized by oily stains that manifests itself under the leaves. Plants susceptible to downy mildew are vine, tomato, potato, lettuce and squash (Zhang et al. 2018; Li et al. 2017). Powdery mildew is characterized by a white powder like a flour that covers the foliage. It affects several plants, but the most sensitive are the oak, the maple, the quince, the apple tree and the hawthorn (Peries 1962; Janisiewicz et al. 2016). The treatment of both types of mildew is the same. Nowadays, farmers are using pesticides to eliminate the majority of diseases including mildew. These pesticides are sprayed by several methods such as manual, by permanent installation or using agricultural autonomous robots.

One of the most important activities of the agriculture sector is the treatment of plants against the disease affecting cultures. Usually, pesticides are used to ensure this treatment, which may have negative side effects on human health and the environment. New methods based on the use of UV-c treatment are developed to treat some diseases such as downy mildew and powdery mildew. Robotic solutions for the implementation of such methods are very interesting, even essential, because of the dangerous effect of UV radiations on the human operators.

In fact, the last decade has known the emergence of robotics in the agricultural field. Many research laboratories and technology providers are working on the development of autonomous vehicles and robots. For instance, the agricultural engineering department at the Louisiana Agricultural Experiment Station developed a robotic seedling transplant model (Hwang and Sistler 1985). The prototype could only transplant at an average rate of six plants per minute, which represents a fifth of the rate for a human operator. The authors have also made a global view of past, present and future agricultural machinery. They identified laboratories interested in agricultural robotics.

Spraying robots and threshing machines are the most famous agricultural robots. Compared to the combine harvester, Sistler (1987) cites several axes that are studied in this context, including the irrigation regulated by robots to minimize water waste during watering. Bonadies et al. (2016), a state of the art is given on unmanned land vehicles (UAVs) used in the field of agriculture to increase efficiency, especially by reducing labor requirements. Other researchers

have developed their robots to improve the harvest of several types of plants such as Van Henten et al. (2002), Sakai et al. (2008) and De-An et al. (2011).

A strawberry harvester was developed and presented in Feng et al. (2012). A manipulator arm with six degrees of freedom with pneumatic gripping fingers and a suction cup was mounted on a four-wheel drive vehicle for harvesting in a greenhouse. The work of Southall et al. (2002) relates to an artificial vision system for an autonomous vehicle designed to treat horticultural crops. The vehicle navigates along rows of crops (individual cauliflower plants) that are planted in a reasonably regular network. The paper of Zhang et al. (2002) gives an overview of the global development of precision farming technologies. This includes the variability of natural resources, variability management, management zone, the impact on the profitability and environment of agricultural holdings, technical innovations in sensors, controls and remote sensing, information management, global applications and adoption trends of precision agriculture technologies.

When spraying, any field location should be treated only once, as excessive distribution of sprayed products will destroy the crop (Janani et al. 2016). On this type of robot, researchers are trying to find the ideal strategies to avoid the destruction of crops with chemical products. Janani et al. (2016), the co-authors propose a cooperative strategy to allow a team of robots to spray on a large field. The goal is to achieve task allocation and coordination using only local information from robots. The proposed strategy is scalable, but requires all robots to participate at the same time. Some reviews like Talbot (2014), Sarri et al. (2017) and Gonzalez-de Soto et al. (2016) were interested in the location of robot with GPS to make an autonomous spray in agricultural field, and to facilitate the movement of robots between the rows of plants without damages. The authors team of Oberti et al. (2016) developed an agricultural robot to detect moisture on plants and apply pesticides to reduce disease on these plants. Using a robotic arm on a wheeled mobile platform and a multi spectral camera, the system can detect the presence of fungi. The vehicle moves and when mildew is detected at a particular position, the robotic arm is used to spray a pesticide on the infected area from three directions to ensure a uniform coverage. The experimental results of this robot revealed an ability to reduce the use of pesticides from 65 to 85%.

The scheduling of robots' tasks in complex agriculture environments is subject to several constraints such as the battery limitation, the evolution of the disease and the duration of treatment. In this article, we address the problem of tasks scheduling on an autonomous mobile robot for the treatment of plants disease in horticulture. We propose a multi-agent based approach to simulate and optimize the treatment missions of the robot while taking into account a limited-capacity rechargeable robot's battery, and a dynamic behaviour of the disease. This work is part of a European project called UV-ROBOT which is intended to use robots that carry type-c ultraviolet (UV-c) lamps to treat plants infected by mildew, in order to replace the chemical treatment. To the best of the authors' knowledge, there is no work in the literature treating the same problem or deploying a similar approach (based on simulation-optimization) to resolve it. The contributions of this paper can be summarized as follows:

- Development of a simulator able to represent the process of mildew treatment by UV-c using a robot, which represents a novelty by itself

- Optimization of the robotized treatment by using three methods:
  - A greedy-based heuristic algorithm;
  - An exact method based on Binary Integer Linear Program (BILP) model;
  - A Genetic algorithm based metaheuristic method.
- Consideration of the dynamic situation, where the level of disease increases with time, using a simulation-optimization approach.

In the rest of this article, we present the steps of our work through the following sections. In the Sect. 2, we review some relevant works related to the treatment of plants diseases, the emergence of robotics in the agriculture sector and the principle methods to simulate and optimize the performances of resulting robotized systems. The Sect. 3 describes the optimization problem of robotized treatment tasks in horticulture and how we formulate it. Then, in Sect. 4, two approximate algorithms are proposed to solve this problem. In Sect. 5, our simulation approach is explained and the development of the simulator is detailed. Then, the hybrid simulation-optimization approach is presented in Sect. 6. In Sect. 7, some experimental results are presented and discussed. Concluding remarks and future works are given in the Sect. 8.

2 Related works

In this paper, we aim to optimize the scheduling of treatment tasks performed by a robot in a greenhouse to reduce the time of treatment, knowing that the robot is running on battery with a limited power capacity. This problem covers several aspects like planning and scheduling of robot's tasks under battery constraint, simulation approach and optimization methods. This section presents some relevant research related to our problem.

In fact, the literature contains more research on robot planning and scheduling in several other areas than the agricultural field. For example, in Brumitt and Stentz (1996), the authors developed a simulator able to plan missions for a fleet of robots. A mission is a set of tasks to be performed by the robot during a predefined period of time, usually between two charging cycles of its battery. When the robots leave their starting point, the simulator can modify their scheduling at any time, which makes them dynamic and more flexible. The obtained results show that each robot is looking for its fastest way to achieve its mission while avoiding obstacles. This problem is NP-hard, even if the authors did not take into account the constraint related to energy capacity of the robots' batteries. Sørensen et al. (2004) worked on agricultural robots tasks planning. The goal was to plan the treatments to be done by robots on a field and to compare them to traditional machine management. The solution proposed is based on graph theory. Based on an aerial image, the field is modeled as an undirected related graph, where each graph edge represents a path. After the construction of the graph, they use a heuristic algorithm based on the Rural Postman Problem (RPP) which allows them to find the shortest path. Dasgupta (2012) summarized his work on multi-robot systems, and emphasized that multi-agent systems (MASs) offer a wide range of solutions that can be adapted to multi-robot systems.

The principle of MAS is to divide a system into multiple agents, such that each agent has its own behaviour in the system.

The objective of the two works presented in Dang et al. (2012, 2014) is to make a task planning for a robot on a finite time horizon, while minimizing the total travel time. The robot transports parts to bins that feed production machines in a warehouse. Dang et al. (2012), the authors developed a GA-based heuristic algorithm. They used a chromosome that contains in each column two variables: the first variable is relative to the machine feed, and the second one is relative to the type of tray to transport. Their algorithm begins to converge towards an optimized solution when the number of generations is greater than 20. Dang et al. (2014), the authors added the mathematical model of the problem and defined time windows for robot feeding tasks based on a (s, Q) inventory policy. It is a classical policy of inventory management, also called 'the reorder point, order quantity' system, where s is the reorder point and Q is the reorder quantity or lot size.

Another interesting work (Giordani et al. 2013) used MAS for multiple robots tasks planning, where the authors modeled the tasks as agents and defined two levels in their system: 'Planning level' and 'Scheduling level'. In planning level, the algorithm assigns a number of robots for each task agent and in each specific period. Then in scheduling level, they use a distributed version of the Hungarian method in order to make a negotiation between the robots. Then, the algorithm makes the calculations and the communication between the robots to assign one robot per task in a given time period.

Several problems could present some similarities with the one considered in this paper, such as the Electrical Vehicle Routing's Problem (EVRP) (Schneider et al. 2014) with a single vehicle, or the Capacitated Vehicle Routing Problem (CVRP) (Laporte and Nobert 1983). The latter seems to be an evident approximation to our problem, but there are several specific characteristics, such as the treatment power consumption, the dynamic level of disease and the battery charging time, which make this approximation complicated to elaborate. The Bin-Packing problem is also often chosen to approximate a big range of problems with different adaptations (Christensen et al. 2017). Other approaches can be used to schedule the robotic tasks such as the coverage path planning (Wei and Isler 2018; Sharma et al. 2019), where a robot must cover/visit several point. However, it is not easy to consider a dynamic variation regarding the importance of each point. Other works on dynamic Bin-Packing problem were studied, where the dynamicity is related to the arrival and departure times of the items. Coffman et al. (1983), the authors have made a natural generalization of the classic Bin-Packing problem. They have used the 'First Fit' (FF) algorithm to manage the arrival and departure times of items dynamically. The works presented in Leinberger et al. (1999), Chan et al. (2009) and Li et al. (2015) aim to minimize the total cost of bins used over time. They used a hybrid algorithm that is based on the FF algorithm. Processing is done on the distribution of requests arising from gaming systems in the cloud. Leinberger et al. (1999) integrated simulation to improve the performance of the FF algorithm for the online Bin-Packing problem. Berndt et al. (2015), the authors studied four cases of packaging problem: Online Bin-Packing, Relaxed Online Bin-Packing, Dynamic Bin-Packing and Fully Dynamic Bin-Packing. In the Fully Dynamic Bin-Packing problem, items arrival and departure happen

in an on-line manner and repackaging of already packaged items is allowed. The goal is to minimize both the number of used bins and the amount of repackaging.

All the aforementioned optimization problems are NP-hard, for which the exact methods are not efficient with big instances. That is why heuristics and meta-heuristics, such as Genetic Algorithm (Karakatič and Podgorelec 2015) or Particle Swarm Optimization (PSO) (Ai and Kachitvichyanukul 2009), are often used to solve this kind of problems, providing a good compromise between the computation time and the quality of the solution.

Several researchers proved the effectiveness of MAS-based simulation. This method gives the possibility to follow the events of the simulation and to make it close to reality. As in Dahane et al. (2017) and Sahnoun et al. (2015), the authors used MAS to predict the health of wind-turbines and to optimize the maintenance of an offshore wind farm. They tested several scenarios in order to obtain the best maintenance strategy.

Other researchers reported that, in many cases, simulation reaches its limits because it does not allow to play certain scenarios where the behaviour of the system changes (Powell et al. 2001; Ören et al. 2014). In order to improve the behaviour of the system or to predict the occurrence of influencing random events, several researchers recommended to add some optimization algorithms into the simulation process (Lim et al. 2009; Powell 2005). In Wu et al. (2003) and Powell (2008), the authors adopted the optimization simulation method and used rough dynamic programming to solve various optimization problems. They applied their method on the problem of the military air planes transport in the United States.

3 Problem formulation

In this work, we consider an autonomous mobile robot that performs the treatment of infected plants in a greenhouse by executing several successive missions. In each mission, it visits a subset of rows containing some infected plants with different levels (see Fig. 1). After each mission the robot must return to the charging station to load its battery before the next mission. Our robot has an average autonomy of 30 min, and its battery loading takes at most 4 h. The capacity limit of the battery is a big challenge for mobile robots. Mei et al. (2005), the authors presented a model of their robots with several graphs showing the energy consumption of different components. For our robot, there are two factors that influence the energy consumption during the execution of a task, which are the speed of the robot's displacement and the UV-c lamps state (on/off).

The appearance and the development of plants' diseases follow different, and probably dependent, stochastic processes. However, for the seek of simplicity, we model this phenomenon by using a simple Markov process, where the transition from a given disease level to the following level is modelled as a Bernoulli trial. In fact, we suppose that each level transition has only two possible outcomes: 'success' (increase of the disease level) or 'failure' (no change). Once a plant is treated, its level of disease is reset to zero. To manage the evolution of diseases in the greenhouse when the robot performs its missions, we turned our 'dynamic problem' into a 24-h time period 'static problem', i.e. the transitions of disease levels are updated each 24 h.

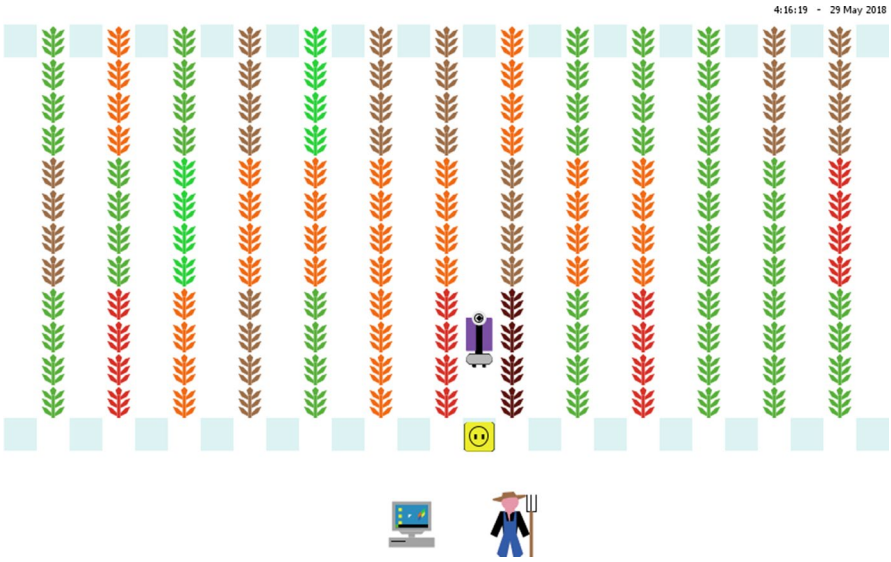


Fig. 1 Representation of greenhouse model with different levels of disease

The 'static problem' consists of scheduling the treatment tasks while minimizing the number of missions needed by the robot to treat all the infected plants in the greenhouse in order to reduce the total time of treatment. The objective is to minimize the impact of disease on the total yield of the greenhouse by eradicating the disease as soon as possible. At the beginning of the planning period, the level of disease of each plant is known. The level of the disease can be assessed either by a visual inspection or by measuring the intensity of mildiou presence in the air close by each plant, using a specific sensor. To treat an infected plant in a given crop, the robot should visit the two surrounding rows to treat the plant from both sides (see Fig. 2). Let w_{ii} be the total amount of energy needed by the robot to treat all infected plants at both sides of the row $i \in \{1, 2, \ldots, N\}$, with N the number of rows in the greenhouse. We assume that the pre-emption of the treatment of infected plants in a given row is not allowed, i.e. each row is visited once and only once. When the robot travels from row i to row j, the amount of power consumed is denoted by w_{ij}. Let W (c.f. Eq. 1) be the power consumption matrix where the principal diagonal elements correspond to the amount of power needed to treat each row in the greenhouse, including the displacement of the robot within the row. The upper and lower diagonal elements of W correspond to the amount of power needed to displace the robot from one row to another. The charging station is considered as a fictive row, indexed by 0, which has no power consumption ($w_{00} = 0$). Note that the charging station corresponds to the starting and ending position of each treatment mission executed by the robot. Let k be the index of missions and X^k its corresponding decision variables matrix, where each element x_{ij}^k is a binary decision variable permitting to assign task (i, j) to mission k. Equation (2) represents an example of missions for a 4-rows greenhouse. During the kth mission, the robot is scheduled to visit rows 1, 3 and 4 successively and then returns back to the charging station.

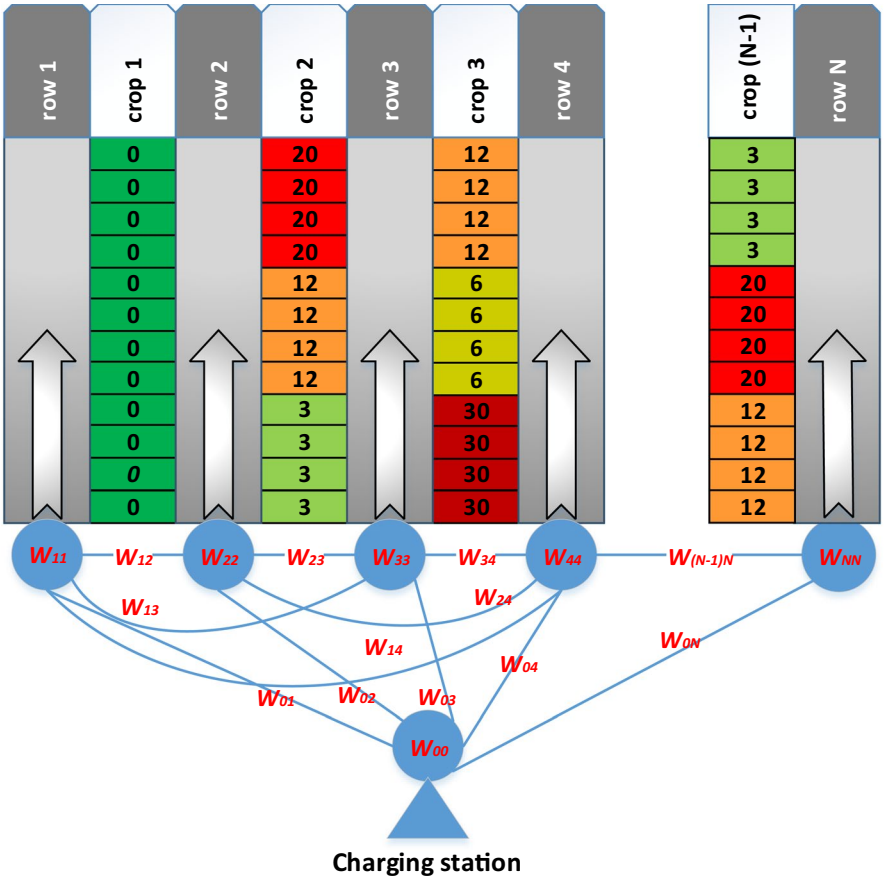


Fig. 2 Diagram of the greenhouse with the tasks assigned to the robot

$$W = \begin{pmatrix} w_{00} & w_{01} & \cdots & w_{0N} \\ w_{10} & w_{11} & \cdots & w_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N0} & w_{N1} & \cdots & w_{NN} \end{pmatrix} \tag{1}$$

$$X^k = \begin{pmatrix} \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 & \mathbf{1} \end{pmatrix} \tag{2}$$

The goal being to treat all infected plants while minimizing the number of missions, this problem is similar to the well known Bin-Packing problem (Mazar et al. 2018). We have the following analogy: items in the Bin-Packing problem correspond to treatment tasks (one task per row visited) and bins correspond to the missions.

Nevertheless, in our case, power consumption during robot displacement should be taken into account, in order to ensure that the robot has the sufficient power to move between rows and to return to the charging station at the end of each mission. For this end, our problem is formulated as a Binary Integer Linear Program (BILP), which is detailed below. Equation (3) represents the objective function, which seeks to minimize the number of missions. In fact, the aim of this study is to optimize the use of the robot. This can be reached by making the same treatment with fewer missions. Reducing the number of missions is equivalent to reducing the total treatment time and a better use of the robot, which allow improving the crop yield.

$$minimize \quad Z(Y) = \sum_{k=1}^{K} y^k \tag{3}$$

Subject to:

$$\sum_{i=0}^{N} \sum_{j=0}^{N} w_{ij} x_{ij}^k \leq C y^k \qquad \forall k \in \{1, ..., K\} \tag{4}$$

$$\sum_{\substack{i=0 \\ i \neq j}}^{N} x_{ij}^k = x_{jj}^k \qquad \forall j \in \{0, .., N\} \quad \forall k \in \{1, .., K\} \tag{5}$$

$$\sum_{\substack{j=0 \\ j \neq i}}^{N} x_{ij}^k = x_{ii}^k \qquad \forall i \in \{0, .., N\} \quad \forall k \in \{1, .., K\} \tag{6}$$

$$y^k \geq y^{k+1} \qquad \forall k \in \{1, .., K-1\} \tag{7}$$

$$\sum_{i=1}^{N} x_{ii}^k \geq y^k \qquad \forall k \in \{1, .., K\} \tag{8}$$

$$x_{ii}^k \leq x_{00}^k \qquad \forall i \in \{1, .., N\} \quad \forall k \in \{1, .., K\} \tag{9}$$

$$\sum_{i=1}^{N} \sum_{j=0}^{i-1} x_{ij}^k = y^k \qquad \forall k \in \{1, .., K\} \tag{10}$$

$$\sum_{k=1}^{K} x_{ii}^k = 1 \qquad \forall i \in \{1, ..., N\} \tag{11}$$

where

- C: Power capacity (in units of power) of the robot's battery at the beginning of the mission
- w_{ij}: Power consumption (in units of power) of the task ij
- x_{ij}^k: a binary decision variable permitting to assign tasks to missions

$$x_{ii}^k = \begin{cases} 1 & \text{if the robot treats row i during mission k} \\ 0 & \text{otherwise.} \end{cases}$$

$$x_{ij}^k = \begin{cases} 1 & \text{if the robot travels directly from row i to row j} \\ 0 & \text{otherwise.} \end{cases} \qquad \forall \quad i \neq j$$

- y^k: a binary decision variable permitting to schedule the missions

$$y^k = \begin{cases} 1 & \text{if the mission k is scheduled} \\ 0 & \text{otherwise.} \end{cases}$$

- K is the maximum number of possible missions. Its upper bound is the number of row of the greenhouse ($K \leq N$). For the execution of the linear program, this value is defined empirically to reduce the number of decision variables.

Constraint (4) ensures that the total energy to be consumed to perform the tasks of each mission k must not exceed the battery's power capacity of the robot. Constraints (5) and (6) define the origin and the destination of a robot when it is visiting a row. It means that the robot has to come from a previously visited row (including the charging station) and it has to visit another row after visiting the current one. Constraint (7) means that mission number $k+1$ can not be scheduled if mission number k is not already scheduled. In constraint (8), if the mission is scheduled, there will be at least one row to visit. Constraint (9) means that no row i can be scheduled if mission number k is not scheduled ($x_{00}^k = 0$). Constraint (10) means that at the end of each mission, the robot goes back to the charging station. Constraint (11) ensures that each row is treated once during one of the scheduled missions.

4 Optimization

This section presents two approximate optimization algorithms developed to solve the 'static problem' formulated in the previous section. These algorithms will be integrated within the simulation process in order to solve the 'dynamic problem', where the stochastic behaviour related to the appearance and the evolution of the disease is taken into account. These algorithms will be evaluated and compared in both static and dynamic environment.

4.1 Heuristic algorithm (HA)

The proposed heuristic (Algorithm 1) is a greedy-based algorithm which assigns treatment tasks to robot missions, iteratively. At the beginning of each iteration, the algorithm initializes the vector of tasks' power consumptions *Vc*, which corresponds to the diagonal of the matrix *W* (line 2), the battery charge *Es* (line 3) and the list of tasks `TASKS=[]` is initialized as empty (line 4). To be sure that the robot can come-back to the charging station, we remove a security power ($Max_k w_{k0}$) corresponding to the power necessary to travel the maximum distance between the charging station

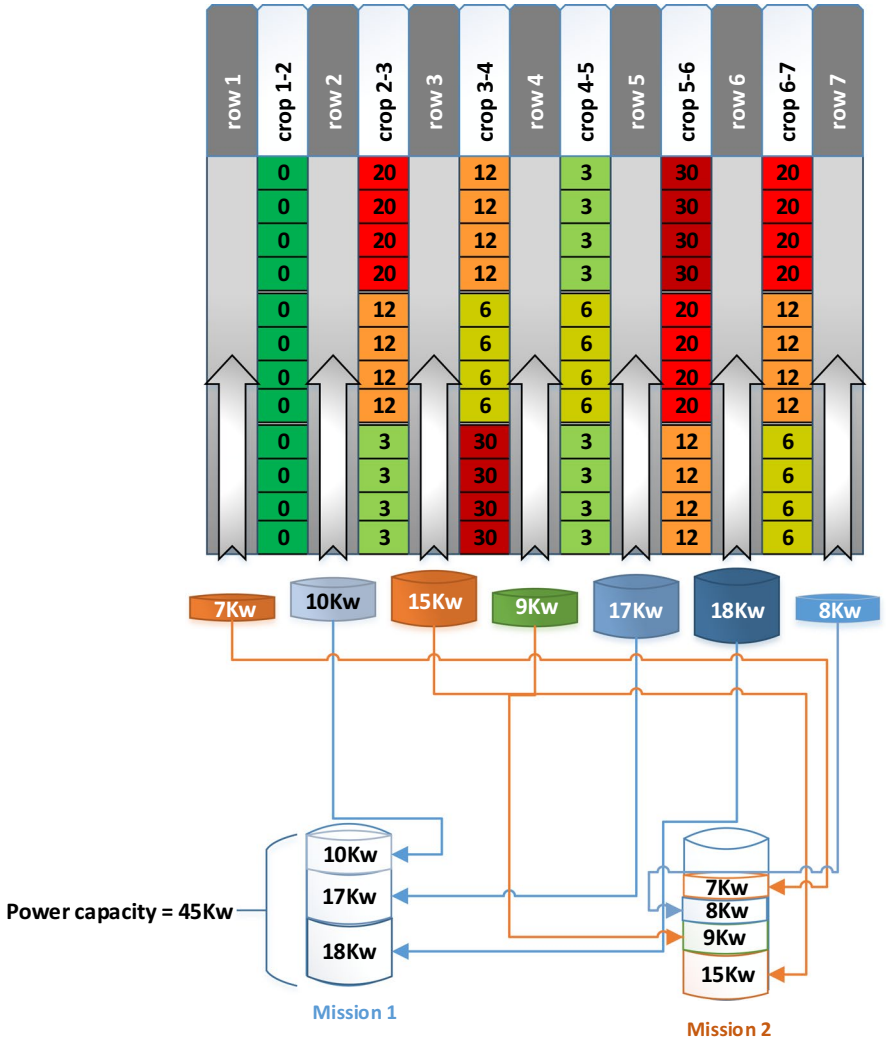


Fig. 3 Mechanism of the heuristic to define the mission of robot

and the farthest row (line 6). The assignment rule to select the tasks of a mission is as follows: the first biggest-power-consumption task that can be appended (lines 8 and 9), i.e. the first task having the biggest power consumption, that is less than or equal to the remaining power capacity minus the power needed to move to its row from the row of the last appended task. If there is enough power, the treatment of selected row is added to the mission (line 11) and the corresponding energy is removed from E (line 12), and this process is continued until testing all the remaining rows.

In order to generate the necessary missions to treat all the greenhouse, this algorithm is repeated several times where the matrix W is updated by not taking into account the tasks already assigned to the previous missions. Figure 3 illustrates an example of the construction of two missions using the heuristic. We can observe that 'Mission 1' contains the biggest tasks (17 kW and 18 kW). The task of 15 kW can not be assigned to this mission because it leads to exceed battery's power capacity of the robot (45 kW), but the task of 10 kW can be treated in this mission. The rest of tasks are assigned to 'Mission 2' using the same algorithm.

Algorithm 1 Treatment tasks assignment heuristic algorithm

Inputs:

W	Updated power consumption matrix
E^0	Initial power level of robot's battery

Outputs:

TASKS	List of tasks assigned to the robot

Variables:

Vc	Power consumption of treatment tasks
CT	Total power consumption in each task
E	Remaining power level of robot's battery

1: **Initialize:**
2: $Vc \leftarrow \boldsymbol{Diag}(W)$
3: $E \leftarrow E^0$
4: TASKS $\leftarrow [\,]$
5: $i \leftarrow 0$
6: $E \leftarrow E - \boldsymbol{Max}_k w_{k0}$
7: **while** $E > 0$ **do**
8: $j \leftarrow \boldsymbol{ArgMax}(Vc)$
9: $CT \leftarrow w_{ij} + Vc(j)$
10: **if** $CT < E$ **then**
11: TASKS $\leftarrow$ TASKS $\cup \{j\}$
12: $E \leftarrow E - CT$
13: $i \leftarrow j$
14: **end if**
15: $Vc(j) \leftarrow 0$
16: **end while**
17: $\boldsymbol{Sort}$(TASKS)
18: **return** TASKS

4.2 Genetic algorithm (GA)

Genetic algorithms represent one of the most used evolutionary solving methods that often perform well approximating solutions to complex problems (Aytug et al. 2003). It is a meta-heuristic optimization algorithm that has the advantage of being quite simple to implement (Tsai et al. 2013). In addition, it is commonly used to resolve the Bin-Packing problem (Falkenauer 1996; Kröger 1995), which is an important motivation to chose GA for the resolution of our problem. However, in order to improve its performances, its parametrization can sometimes become a delicate task. Even if there are several rules to follow in order to define the GA parameters, each problem has its own characteristics and needs an empirical adaptation of the GA parameters. The chromosome coding and the GA operators are detailed below. The fitness function corresponds to the total number of missions needed to assign all treatment tasks.

The chromosome of the GA is coded as a binary matrix where lines represent missions and columns represent the greenhouse's rows. For example, if the robot has to treat row j during the mission i, the gene (i, j) gets a value of one, zero otherwise.

At the beginning, an initial population of 50 individuals is created randomly. To create the population of a new generation, ordinary genetic operators are successively applied, namely: '*Crossover*', '*Mutation*' and '*Selection*'. Each created chromosome, either in the initial population creation or by genetic operators, is tested by verifying the constraints developed above so that all the chromosomes are feasible.

Each run of the GA is executed as follows: two chromosomes are selected randomly from the current population and crossed. Each generated child is tested and is regenerated until it becomes feasible. The obtained children are then mutated and validated again. For our algorithm, the average rate of infeasible generated children was around 7%, which indicates that the diversity of generated chromosome is ensured (Abdelaziz et al. 1999). If this rate increases, it will increase the computation time, but if it is null, that mean that there is a risk of non diversity of the population (Aickelin and Dowsland 2004). All the parameters of the GA are defined empirically after several trails.

4.2.1 Crossover

Figure 4 illustrates the structure of the chromosomes and the principle of the crossover operation. It is a single-point crossover, which is randomly generated between 2 and $N - 1$. 90% of the population is randomly selected for the crossover operation. We opted for random selection to reduce the calculation time. Although there are several ways in the literature to select parents, such as the 'roulette wheel', We use a simpler selection method because we did not meet any phenomena of loss of diversity.

4.2.2 Mutation

Obtained children may be mutated with a given probability according to the following four possibilities: (1) only the first child is mutated, with probability 0.3, (2)

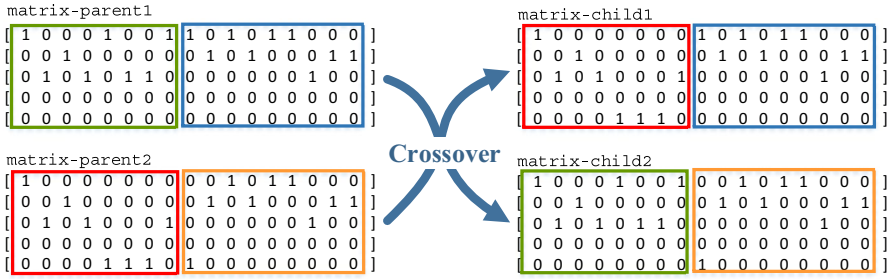


Fig. 4 Illustration of the crossover operator

only the second child is mutated, with probability 0.3, (3) both children are mutated, with probability 0.3, and (4) neither children are mutated, with probability 0.1. The mutation operation is illustrated in Fig. 5. Two rows of the chromosome (missions) are randomly selected and their elements are respectively interchanged one by one, following a Bernoulli process with probability $P = 0.5$.

Each chromosome from the new population is evaluated and discarded if it is infeasible, if it does not respect at least one constraint.

4.2.3 Selection

10% of the best chromosomes of the old generation are selected to be directly a part of the new generation. 90% of this new generation are selected from the best individuals obtained by crossover and mutation.

4.2.4 Stopping test

This process is stopped after a fixed number of generations. The size of the population and the maximum generation can be defined manually using a slider on the graphical interface of the sim-optimizer. For the results presented here, we set this number to 20 after several trials. The best individual, having the minimum number of missions to treat all infected rows, is returned.

4.3 Exact method

The developed BILP (detailed in Sect. 3) has been solved using a commercial solver 'FICO® Xpress' to give optimal solution of the problem. In order to reduce the number of decision variables, the maximum number of mission is defined, for each instance, by the number of mission given by the heuristic algorithm.

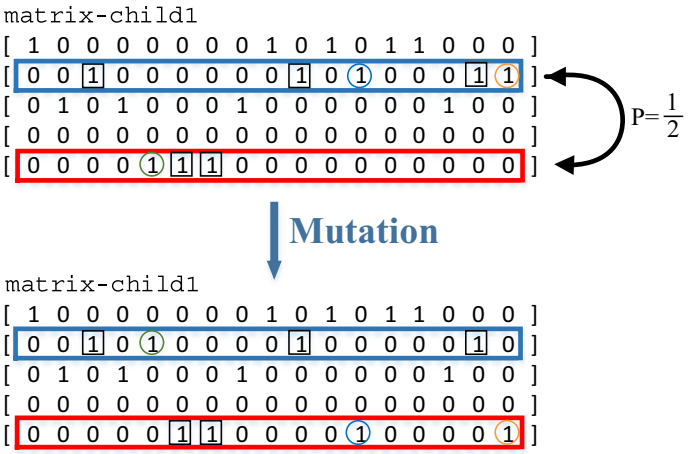


Fig. 5 Mutation method

5 Simulation

The dynamic and stochastic behaviour of the apparition and evolution of the disease in the greenhouse changes the problem during time. This behaviour can not be included in the optimization model proposed above. Changing the parameters of the model means the resolution of another problem completely different. The technique of simulation and optimization can be used to consider this dynamicity of the system. This section describes the development of the simulator. Since the considered system used by the UV-Robot is complex, we chose to use the MASs for its modeling and simulation. MAS allows the representation of each agent interdependently and facilitates, by the way, the modeling and simulation of complex systems. When modeling the system by MAS, its is important to divide the systems into agents to allow their modeling and the definition of their interactions.

Figure 6 presents the simulation model using MASs, were there are 7 agents and 10 interactions between them. The agents are defined as follow:

- *Grower* its role consists to setup the robot and repair or manually transport it to the recharge station when there is a problem. We consider that the grower plays the role of supervisor, who should be always present.
- *Robot* only one robot agent is considered, which is able to execute autonomously a set of missions defined and scheduled by the monitoring agent. It controls its speed and the state of UV-lamps regarding the state of plants (the level of disease). The robot have a limited electric power capacity that decreases according to its speed and the state of UV-Lamps. Figure 7 represents the behaviour of the robot during the execution of treatment missions in the greenhouse. In fact, each mission is composed of the treatment of K_{max} rows. Each row is composed of J_{max} different sections of plants (each section can be of 1–4 m). The treatment of each row starts by going to the entry of the row. Then, the robot treats all the

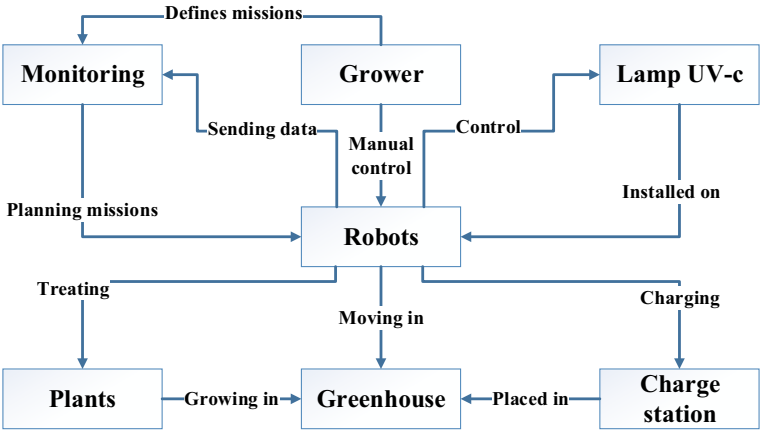


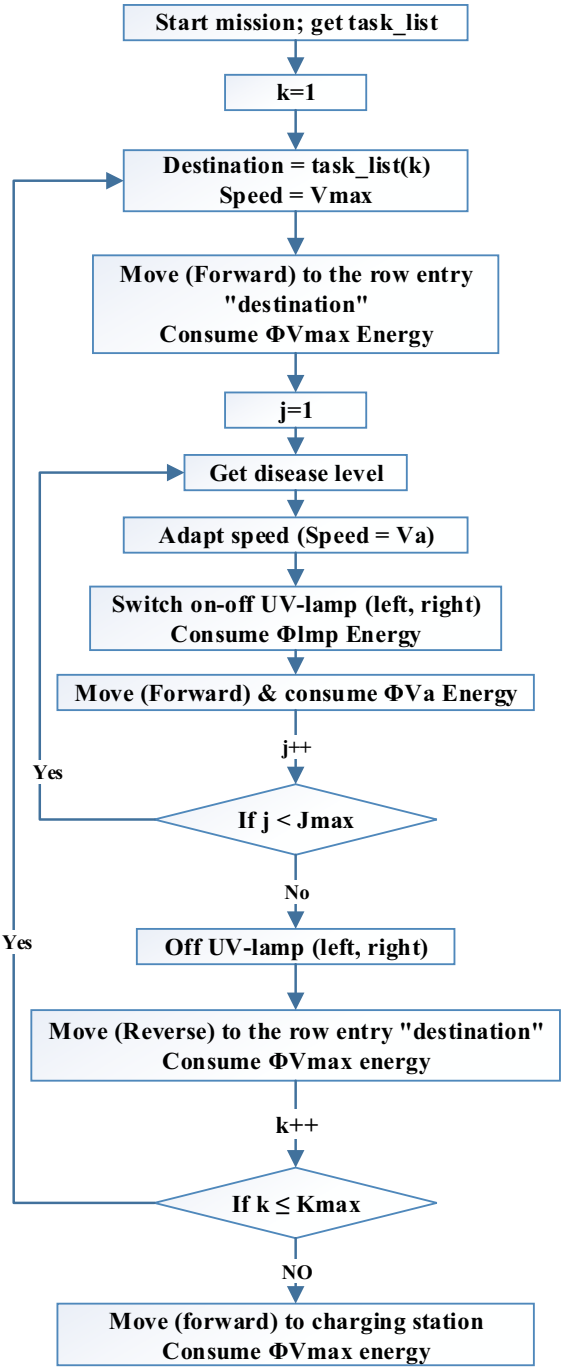
Fig. 6 Multi agent system model

sections successively, while adapting its speed (V_a) and the state of the lamps accordingly to the level of disease of each section. When the robot arrives to the end of the row ($j = J_{max}$), it switches off the UV-lamps and goes back to the entry of the row with the maximum speed V_{max}. The energy consumption of each action is defined by the following values: ΦV_{max}, corresponding to robot displacement with maximum speed; ΦV_a, corresponding to robot displacement with the adapted speed V_a; and $\Phi lamps$, the energy consumption of the UV-lamps.

- *UV-lamps* this agent is placed on the robot and controlled by it.
- *Plants* the plants agents are able to grow and degrade their situation when they are affected by disease. The disease level in each agent *Plant* is a stochastic process that is influenced by other plants, the environment and the state of the plant itself. After treatment, the level of disease of the plant is set to zero. Six levels of disease are considered: the plant is safe if the level is zero and completely infected if the level is 30. The apparition and evolution of the disease in the greenhouse is supposed to follow a Markovian process. The transition probabilities are defined by the user at the beginning of the simulation process.

  A plant is considered as fully treated only when the robot treats it from both sides (left, right), and its disease level is reset to zero. There is also in the agent *Plant* a function called *Plants state*, which checks the state of the plant before producing the fruit. The relation between the level of disease and the production is inversely proportional.
- *Greenhouse* this agent represents the environment in which the other agents are evolving. General indicators are related to this agent such as the global level of disease.
- *Charge station* this agent manages the charging operation of the robot's battery. When the robot visits the charging station, its battery becomes fully charged after a time duration that depends on the initial power level of the battery at the beginning of the charging operation. In our case, if the robot's battery is completely empty, the charging duration is about 4 h.

Fig. 7 Robot behaviour



- *Monitoring* this agent monitors the system and defines the missions for the robot. The monitoring function includes the observation of the level of mildew, the environment and state of the robot (position, charge, health level, etc.). Based on this information, the agent *Monitoring* makes decision by optimizing the mission for the robot using one of the algorithms detailed in Sect. 4. The optimization part is ensured by the function 'Optimize the mission planning' as shown in Fig. 8. The selection of the optimization method is done manually before the beginning of the simulation process. The moment of running the optimization algorithm during the simulation is explained in Sect. 6.

The interactions between agents are defined as follows:

The agent *Grower* starts the process of treatment by launching the agent *Monitoring*. Then, the agent *Monitoring* collects data from the agents *Robot* and *Grower* and runs an optimization algorithm to schedule the missions for the agent *Robot*. After this step, the *Robot* starts the treatments by moving in the *Greenhouse* and visiting the *Plants* growing in the *Greenhouse*. The *Robot* sends its position and its battery's remaining power level to the monitoring system. Even if there is no direct link between the agent *Monitoring* and the agent *Greenhouse*, the *Robot* is playing the role of communication channel between these two agents. We assume that the agent *Greenhouse* is able to know the levels of disease of all plants and send them automatically to the agent *Monitoring*. The treatment of each plant section is done by turning on or off the *UV-lamps* that are installed on the *Robot*. After each mission the agent *Robot* goes to the *ChargeStation* which is placed in the *Greenhouse*.

The above MAS model has been used to develop our simulator with Netlogo software (Wilensky and Evanston 1999). The simulator allows the representation of the

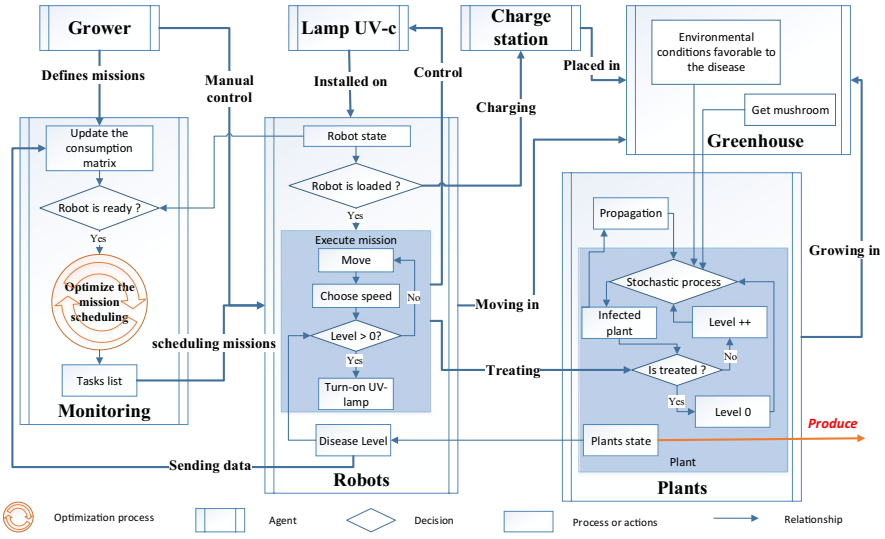


Fig. 8 Agent-based simulation optimization process

behaviour of our system, including the visualization of the infection level evolution on each plant section in the greenhouse.

Generally, the simulation process allows the execution of a limited number of scenarios and compare their results. The user can make decision based on these observations. This manner of decision-making allows to obtain feasible solution, but there is not a real exploration of the state space of system. The agent *Monitoring*, is able to make complex decisions because it includes some optimization algorithm. The execution of this algorithm during the simulation can improve the behaviour of the system and the value of its key performance indicators. Section 6 explains how to integrate the optimization process into the simulation.

6 Simulation-optimization

Simulation and optimization are the most important methods used for decision making. Simulation gives a vision of the process in time (exploration of future state), but its vision in the space state is limited (limited exploration of all the possible choices for a treatment). On another hand, the simulation can explore the space state but its vision in time is limited. In fact, it is not obvious to consider the variation of system parameters. Mixing these two techniques by using simulation-optimization approach can resolve this problem. The idea is to optimize simulation problems over time, by making decisions that takes into account the future situation of the system, Powell (2008) or by exploring the state space through the stochastic behaviour of the system (Wu et al. 2003). In order to integrate this decision in the simulation, we have to answer several questions such as: (1) how to introduce the optimization algorithms and for which parameter? (2) when should the optimization algorithm be launched during the simulation process? and (3) what is the horizon of optimization?

The agent *Monitoring is* responsible for defining the mission for the agent *Robot*. The decision to schedule a mission can be defined by one of the optimization methods presented above, or simply by using the numerical order of the rows. In fact, the agent *Monitoring* collects data and receives orders from several agents in the greenhouse. It receives the order to start optimizing the mission from the grower and it receives the information concerning the level of disease and the state of the robot from the agent *Robot*, as shown Fig. 8. The list of treatments to execute during a mission is transmitted to robot after the optimization process. The selection of the optimization method is defined manually by the Grower (user) and the moment the optimization process is launched depends on the selected algorithm and the data collected from the robot. For example, the heuristic is launched after the end of each mission, but the GA is launched after the treatment of all the greenhouse.

Since, each mission is a set of rows to visit, and the optimization process defines these rows in the aim to reduce the number of missions. The optimization process can be executed just before each mission (heuristic case) or when the robot finish all the scheduled missions without eliminating all the disease in the greenhouse (GA, exact method with dynamic disease evolution process). The parameters that influence the launching of an optimization process can be summarized as follows:

- The power level of the robot's battery
- The position of the robot
- The level of disease
- The type of selected optimization algorithm.

Figure 8 shows the decision process in the simulation model for each agent. The optimization algorithm is a part of the agent *Monitoring*, which receives the plants' disease levels from the robot in order to update the values of the diagonal elements ($w_{i,i}, i = 0\ to\ N$) of energy consumption matrix W (c.f. Eq. 1). Figure 8 also contains more details about the other agents (*Robot*, *Plants* and *Greenhouse*), their inner decision processes and their interactions. As soon as the agent *Robot* receives its mission, it begins executing it by moving between the rows selected within the mission. The list of selected rows is generated by the optimization process and transmitted by the agent *Monitoring*. The robot treats infected plants using UV-c lamps by adapting its speed according to the disease level. The lamps are switched off during the displacement between rows or in front of healthy plant.

The optimization algorithms are managed through the interface of the simulator, where the user chooses the appropriate algorithm before launching the simulation. Then, the monitoring makes the decisions using only the selected algorithm. We notice that, whatever the selected algorithm, the greenhouse will be treated until all plants' diseases are totally eradicated.

Three different algorithms were proposed to optimize this process in static and dynamic situations. The next section will present the tests performed to test the efficiency of proposed algorithms regarding the CPU time and the objective function quality.

7 Experimentation

The aim of this section is to present and compare the results obtained by each proposed optimization algorithm. The first phase of tests is dedicated to evaluate the average gap (GAP) between the solutions provided by the GA and HA compared to the optimal solutions provided by the exact method (EM). The second phase of tests is dedicated to the test of the performance of these methods in the case of a dynamic system, where the parameters of the model are changing over time (variation of the level of disease). The solution based on the sim-optimization method is then proposed to deal with the dynamic behaviour of the disease.

7.1 Phase 1: static environment

In order to test the developed model, algorithm and simulator, a set of simulation optimization experiments were performed.

To evaluate the results obtained by HA, we compared them with those of EM obtained by a commercial solver, namely the 'FICO® Xpress Workbench' solver. The obtained results (Fig. 9) show that the heuristic solution is very close to the

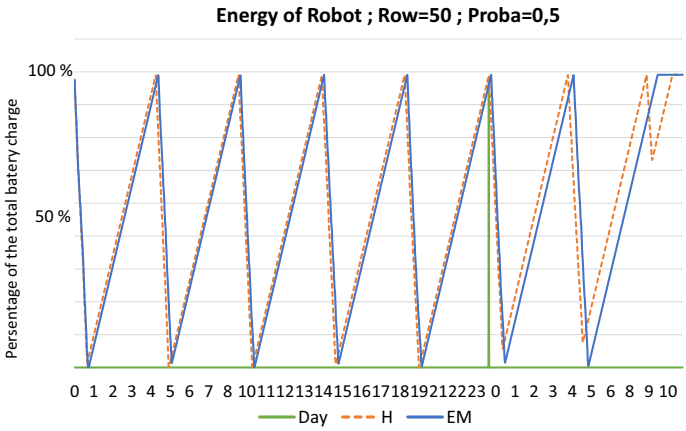


Fig. 9 Robot energy consumption with heuristic and exact method

optimal solution. Figure 9 draws the robot's energy consumption for the treatment of a greenhouse composed of 50 plant rows, where the disease's probability of apparition P is equal to 0.5. In both curves (exact method and heuristic), the robot uses a battery of 960 Wh of energy capacity, allowing it to execute each mission during around 30 min before its battery is being charged during around 4 h. The treatment of all infected rows is carried out in 7 missions with EM and 8 missions using HA. Figure 9 shows also that there is a tiny difference between both methods in the first five missions. However, in the two last missions, HA does not allow the robot to use all of the available energy on its battery. This can be explained by the fact that, in the two last missions, HA cannot find any mission that can be executed using the remaining energy. In the same time, the optimal solution uses all the available energy during each mission. The total treatment with EM consumes about 2% less energy than the heuristic method and finishes the treatment 3 h and 40 min earlier. Based on this observations, we can conclude that the heuristic can be a good alternative regarding its execution time and solution quality.

After the validation of the results obtained by HA, GA was tested for several greenhouse sizes with different disease's probabilities of apparition. We compared the three methods for each greenhouse. Table 1 summarizes 540 simulation runs for 9 different greenhouses configurations, where 20 simulations are performed for each one. We choose imperially to perform 20 experiments to get a realistic stable average. For GA and HA, the average and the standard deviation of their gap compared to EM are presented in the second and the third columns, respectively. The column '# Non Convergence' represents the number of simulations, out of 20, where EM did not converge in a reasonable time. We consider that there is no convergence if the CPU time exceeds 8 h without any result. In fact, due to the NP-hardness of the problem, the exact method can not always converge with rematively large instances ($R = 75$ and $P = 1$; $R = 100$ and $P = 0.75$; $R = 100$ and $P = 1$). For both approximate algorithms, the gaps are near to zero, which means that the obtained results are not very far from the optimal solutions. The comparison of the gaps of HA and

GA demonstrates that the latter gives better solutions in three cases (presented with boldface in the column GA GAP). The comparison of the standard deviation (SD) demonstrates the stability of the obtained results for each method.

Table 2 presents the average and the standard deviation of CPU time for each used algorithm. Results show that the CPU time is increasing with the instance size, as well as the standard deviation. For example, in the case of large instance ($R = 100$ and $P = 1$), the average CPU time is 10,426 s (2 h, 55 min and 24 s) for EM method, 11.075 s for GA and 0.084 s for HA. For smallest instances, these times are 7.75 s for EM, 1.448 s for GA and 0.016 s for HA. For all the tested instances, it is clear that HA is faster than GA, which is faster than EM. The values of the standard deviation demonstrate that methods are stable and the recorded CPU times are varying in a small range. We notice that this time can be influenced by other programs executed in the same time by the computer, like anti-virus or other hidden services of the operating system.

In order to understand the evolution of the disease during the treatment, we present the disease level using each algorithm in Fig. 10 for the case of a large instance ($R = 100$ and $P = 0.5$). The total treatment of the greenhouse takes more than 2 days for all algorithms. EM allows to finish first (blue curve) within 14 missions, whereas GA allows to finish the treatment within 15 missions (doted red curve) and HA within 16 missions. Each vertical green line indicates the beginning of a calender day. The 4-h periods of time where the level of disease is constant correspond to charging cycles. The periods of time where the level of disease decreases correspond to treatment cycles. The pace of decreasing is low in the first missions (missions 1–4) because the robot treats a lot of rows from only one side, whereas a row is considered as treated only when the treatment is performed from its both sides.

To sum up about this part of experiments, we can conclude that the proposed HA and GA present interesting performances in terms of processing time and solution quality. Moreover, GA has the advantage of improving solution quality compared to HA, but it consumes insignificantly more CPU time.

Table 1 GAP average and standard deviation for the three algorithms for different values of R and P (bold → GA performs better than HA)

(R, P)	Heuristic GAP/SD	Genetic algorithm GAP/SD	Exact method # Non convergence
(50, 0.5)	0.056/0.062	**0.044/0.061**	0
(50, 0.75)	0.022/0.038	0.022/0.039	0
(50, 1)	0.016/0.22	0.016/0.21	1
(75, 0.5)	0.054/0.045	0.054/0.042	0
(75, 0.75)	0.049/0.021	0.049/0.025	0
(75, 1)	0.021/0.022	0.021/0.021	14
(100, 0.5)	0.041/0.033	**0.037/0.034**	2
(100, 0.75)	0.051/0.016	**0.038/0.015**	13
(100, 1)	0.023/0.015	0.023/0.014	16

Table 2 CPU time average and standard deviation for the three algorithms for different values of R and P

(R, P)	Heuristic CPU time/SD (s)	Genetic algorithm CPU time/SD (s)	Exact method CPU time/SD (s)
(50, 0.5)	0.016/0.015	1.448/0.09	7.75/2.13
(50, 0.75)	0.010/0.014	3.73/0.19	18/7.1
(50, 1)	0.016/0.009	6.153/0.89	73/49
(75, 0.5)	0.046/0.097	3.172/0.75	30/64.7
(75, 0.75)	0.029/0.004	6.563/0.39	217/94
(75, 1)	0.038/0.013	9.994/1.9	1677/884
(100, 0.5)	0.049/0.006	5.253/0.74	82/119
(100, 0.75)	0.058/0.016	11.751/2.27	431.286/21334
(100, 1)	0.084/0.034	11.075/2.9	10426/81203

7.2 Phase 2: dynamic environment

In practice, as explained in Sect. 3, the evolution of the disease happens all the time but it is modeled as 24-h static problem in the developed simulation model. In order to test the performances of the proposed algorithms (HA, GA and EM), we consider the same instance that was tested for the static environment case ($R = 100$ and $P = 0.5$). This instance is selected for this test of the dynamic case, because it is the largest instance that EM can solve in reasonable time. The results are reported in Fig. 11, which shows the evolution of the total level of disease in the greenhouse, for the three algorithms, until it is totally treated. As it can be seen, the level of disease is updated (increases) at the end of each day (green line).

HA is executed at the beginning of each mission, whereas GA and EM are executed at the end of all scheduled missions for one equivalent Bin-Packing problem. Because HA generates tasks scheduling of only one mission, the simulator waits until the end of the current mission to update the level of disease and launch

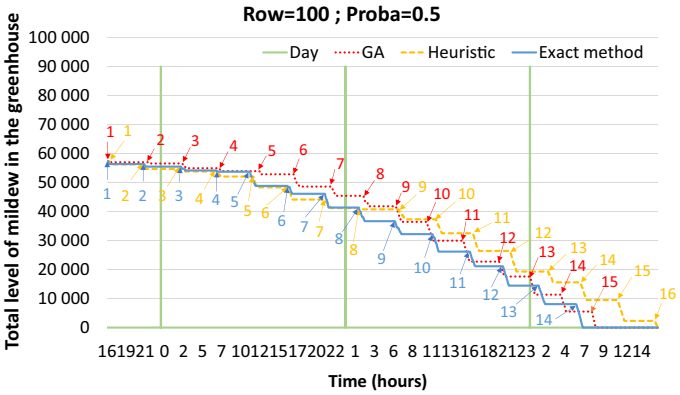


Fig. 10 Level of mildew in static environment

to define the following mission in a negligible computation time. Concerning GA and EM, as both methods generate a set of missions, they are launched at the end of all planned missions. In this case, the unique way to compare HA with the two other methods is to execute it until treating all the greenhouse. The execution moments of GA and EM are mentioned in Fig. 11 by GA_i and EM_i respectively, where $i \in \{1, 2, 3\}$ corresponds to the number of algorithm execution. The total treatment time of the greenhouse using EM and GA is about 4 days (3 days in the static case), whereas it is about 6 days for HA. GA increases the total treatment time by 4 h (only one additional mission) compared to EM (c.f. Fig. 11), which is a very interesting approach regarding its small computation time (5 s for GA and 82 s for EM). This time increases sharply with bigger instances (Table 2). In addition, the solution given by GA can be obtained in an on line time. This time is limited by the necessary time for a full charging of the robot, which is in a constant evolution. To conclude, the efficiency of HA decreases in the dynamic environment case, while GA still presents a very interesting results, which are close to the optimal solution provided by EM.

8 Conclusion

In this paper, a simulation-optimization approach has been used to solve the problem of robotized tasks scheduling for mildew treatment by UV-c rays in horticulture. The problem has been formulated as a classical Bin-Packing problem and a simulator has been developed using the paradigms of multi-agent systems to track system events and behaviour in static and dynamic environments. Three optimization algorithms (HA, GA and EM) have been introduced into the simulation process to improve the decision making process of the system. The merging of optimization and simulation involves making reliable decisions about

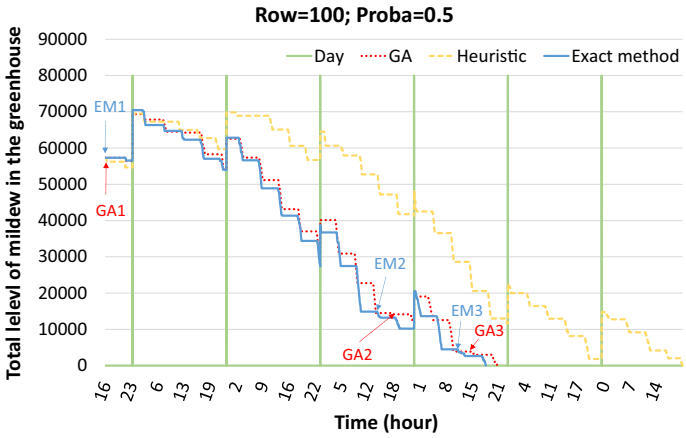


Fig. 11 Level of mildew in dynamic environment

the timing of the optimization algorithms in the simulator. During the simulation process, HA is launched at the beginning of each mission, whereas GA and EM are launched only once in the static case and at the end of a set of planned missions in the dynamic environment case. A set of experimentations have been conduced to compare the performances of the proposed optimization algorithms. The obtained results show that HA is efficient in the static environment case, but its performance is degraded in the dynamic environment case. GA presents very interesting performances in both cases (dynamic and static), especially with large instances. Regarding the execution time of GA and the good quality of its results, we recommend the use of GA, which can deal with big instance in short time.

The simulation-optimization method manages the dynamic behaviour of complex systems and GA is very interesting method in these cases.

Our next work will focus on the development of methods for the totally-dynamic environment case. In this case, the constraint of static state periods of 24 h will be removed, which means that the infection by the diseases is continuously increasing during the whole simulation period. The next step in this project is the deployment of the proposed method in a real case with the robot developed by our partners in the project. Another perspective of this works is to study the case of multi-robots, multi-charging-stations and multi-greenhouses.

Acknowledgements This research was made possible thanks to €1.35 million financial support from the European Regional Development Fund provided by the Interreg North-West Europe Programme in context of UV-ROBOT.

References

Abdelaziz FB, Krichen S, Chaouachi J (1999) A hybrid heuristic for multiobjective knapsack problems. Meta-heuristics. Springer, Boston, pp 205–212

Ai TJ, Kachitvichyanukul V (2009) A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. Comput Oper Res 36(5):1693–1702

Aickelin U, Dowsland KA (2004) An indirect genetic algorithm for a nurse-scheduling problem. Comput Oper Res 31(5):761–778

Aytug H, Khouja M, Vergara F (2003) Use of genetic algorithms to solve production and operations management problems: a review. Int J Prod Res 41(17):3955–4009

Berndt S, Jansen K, Klein KM (2015) Fully dynamic bin packing revisited. Math Program. https://doi.org/10.1007/s10107-018-1325-x

Bonadies S, Lefcourt A, Gadsden SA (2016) A survey of unmanned ground vehicles with applications to agricultural and environmental sensing. In: Autonomous air and ground sensing systems for agricultural optimization and phenotyping, vol 9866. International Society for Optics and Photonics, p 98660Q

Brumitt BL, Stentz A (1996) Dynamic mission planning for multiple mobile robots. In: Proceedings IEEE international conference on robotics and automation, 1996, vol 3. IEEE, pp 2396–2401

Chan JWT, Wong PW, Yung FC (2009) On dynamic bin packing: an improved lower bound and resource augmentation analysis. Algorithmica 53(2):172–206

Christensen HI, Khan A, Pokutta S, Tetali P (2017) Approximation and online algorithms for multidimensional bin packing: a survey. Comput Sci Rev 24:63–79

Coffman EG Jr, Garey MR, Johnson DS (1983) Dynamic bin packing. SIAM J Comput 12(2):227–258

Dahane M, Sahnoun M, Bettayeb B, Baudry D, Boudhar H (2017) Impact of spare parts remanufacturing on the operation and maintenance performance of offshore wind turbines: a multi-agent approach. J Intell Manuf 28(7):1531–1549

Dang QV, Nielsen IE, Bocewicz G (2012) A genetic algorithm-based heuristic for part-feeding mobile robot scheduling problem. In: Trends in practical applications of agents and multiagent systems. Springer, Berlin, pp 85–92

Dang QV, Nielsen I, Steger-Jensen K, Madsen O (2014) Scheduling a single mobile robot for part-feeding tasks of production lines. J Intell Manuf 25(6):1271–1287

Dasgupta P (2012) Multi-agent coordination techniques for multi-robot task allocation and multi-robot area coverage. In: 2012 international conference on collaboration technologies and systems (cts). IEEE, pp 75–75

De-An Z, Jidong L, Wei J, Ying Z, Yu C (2011) Design and control of an apple harvesting robot. Biosyst Eng 110(2):112–122

Falkenauer E (1996) A hybrid grouping genetic algorithm for bin packing. J Heuristics 2(1):5–30

Feng Q, Wang X, Zheng W, Qiu Q, Jiang K (2012) New strawberry harvesting robot for elevated-trough culture. Int J Agric Biol Eng 5(2):1–8

Giordani S, Lujak M, Martinelli F (2013) A distributed multi-agent production planning and scheduling framework for mobile robots. Comput Ind Eng 64(1):19–30

Gonzalez-de Soto M, Emmi L, Perez-Ruiz M, Aguera J, Gonzalez-de Santos P (2016) Autonomous systems for precise spraying-evaluation of a robotised patch sprayer. Biosyst Eng 146:165–182

Hwang H, Sistler F (1985) The implementation of a robotic manipulator on a pepper transplanting machine. In: Proceedings of the international conference on CAD/CAM, robotics automation, pp 553–556

Janani A, Alboul L, Penders J (2016) Multi robot cooperative area coverage, case study: spraying. In: Conference towards autonomous robotic systems. Springer, pp 165–176

Janisiewicz WJ, Takeda F, Nichols B, Glenn DM, Jurick WM II, Camp MJ (2016) Use of low-dose UV-C irradiation to control powdery mildew caused by *Podosphaera aphanis* on strawberry plants. Can J Plant Pathol 38(4):430–439

Karakatič S, Podgorelec V (2015) A survey of genetic algorithms for solving multi depot vehicle routing problem. Appl Soft Comput 27:519–532

Kröger B (1995) Guillotineable bin packing: a genetic approach. Eur J Oper Res 84(3):645–661

Laporte G, Nobert Y (1983) A branch and bound algorithm for the capacitated vehicle routing problem. Oper Res Spektrum 5(2):77–85

Leinberger W, Karypis G, Kumar V (1999) Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints. In: Proceedings of the 1999 international conference on parallel processing. IEEE, pp 404–412

Li Y, Tang X, Cai W (2015) Dynamic bin packing for on-demand cloud resource allocation. IEEE Trans Parallel Distrib Syst 27(1):157–170

Li J, Wang P, Geng C (2017) The disease assessment of cucumber downy mildew based on image processing. In: 2017 international conference on computer network, electronic and automation (ICCNEA). IEEE, pp 480–485

Lim MK, Zhang Z, Goh W (2009) An iterative agent bidding mechanism for responsive manufacturing. Eng Appl Artif Intell 22(7):1068–1079

Mazar M, Sahnoun M, Bettayeb B, Klement N (2018) Optimization of robotized tasks for the UV-C treatment of diseases in horticulture

Mei Y, Lu YH, Hu YC, Lee CG (2005) A case study of mobile robot's energy consumption and conservation techniques. In: Proceedings 12th international conference on advanced robotics, 2005. ICAR'05. IEEE, pp 492–497 (2005)

Oberti R, Marchi M, Tirelli P, Calcante A, Iriti M, Tona E, Hočevar M, Baur J, Pfaff J, Schütz C et al (2016) Selective spraying of grapevines for disease control using a modular agricultural robot. Biosyst Eng 146:203–215

Ören T, Yilmaz L, Ghasem-Aghaee N (2014) A systematic view of agent-supported simulation past, present, and promising future. In: 2014 international conference on simulation and modeling methodologies, technologies and applications (SIMULTECH). IEEE, pp 497–506

Peries O (1962) Studies on strawberry mildew, caused by *Sphaerotheca macularis* (wallr. ex fries) jaczewski. Ann Appl Biol 50(2):211–224

Powell WB (2005) The optimizing-simulator: merging simulation and optimization using approximate dynamic programming. In: Proceedings of the 37th conference on winter simulation. Winter Simulation Conference, pp 96–109 (2005)
Powell WB (2008) Approximate dynamic programming: lessons from the field. In: Simulation conference, 2008. WSC 2008, Winter. IEEE, pp 205–214
Powell WB, Shapiro JA, Simao HP (2001) A representational paradigm for dynamic resource transformation problems. Ann Oper Res 104(1):231–279
Sahnoun M, Baudry D, Mustafee N, Louis A, Smart PA, Godsiff P, Mazari B (2015) Modelling and simulation of operation and maintenance strategy for offshore wind farms based on multi-agent system. J Intell Manuf 30(8):2981–2997
Sakai S, Iida M, Osuka K, Umeda M (2008) Design and control of a heavy material handling manipulator for agricultural robots. Auton Robots 25(3):189–204
Sarri D, Martelloni L, Vieri M (2017) Development of a prototype of telemetry system for monitoring the spraying operation in vineyards. Comput Electron Agric 142:248–259
Schneider M, Stenger A, Goeke D (2014) The electric vehicle-routing problem with time windows and recharging stations. Transp Sci 48(4):500–520
Sharma G, Dutta A, Kim JH (2019) Optimal online coverage path planning with energy constraints. In: Proceedings of the 18th international conference on autonomous agents and multiagent aystems. International Foundation for Autonomous Agents and Multiagent Systems, pp 1189–1197
Sistler F (1987) Robotics and intelligent machines in agriculture. IEEE J Robot Autom 3(1):3–6
Sørensen C, Bak T, Jørgensen R (2004) Mission planner for agricultural robotics. AgEng 2004:894–895
Southall B, Hague T, Marchant JA, Buxton BF (2002) An autonomous crop treatment robot: part I. A kalman filter model for localization and crop/weed classification. Int J Robot Res 21(1):61–74
Talbot D (2014) A nimble-wheeled farm robot goes to work in Minnesota, MIT Technology Review, 9 September 2014. [Online]. Available: https://www.technologyreview.com/s/530526/a-nimble-wheeledfarm-robot-goes-to-work-in-minnesota/. Accessed 18 Dec 2019
Tsai CF, Eberle W, Chu CY (2013) Genetic algorithms in feature and instance selection. Knowl-Based Syst 39:240–247
Van Henten EJ, Hemming J, Van Tuijl B, Kornet J, Meuleman J, Bontsema J, Van Os E (2002) An autonomous robot for harvesting cucumbers in greenhouses. Auton Robots 13(3):241–258
Wei M, Isler V (2018) Coverage path planning under the energy constraint. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 368–373
Wilensky U, Evanston I (1999) Netlogo: center for connected learning and computer-based modeling. Northwestern University, Evanston, pp 49–52
Wu T, Powell WB, Whisman A (2003) The optimizing simulator: an intelligent analysis tool for the military airlift problem. Unpublished report. Department of Operations Research and Financial Engineering, Princeton University, Princeton
Zhang N, Wang M, Wang N (2002) Precision agriculture-a worldwide overview. Comput Electron Agric 36(2–3):113–132
Zhang S, Ding F, Peng H, Huang Y, Lu J (2018) Molecular cloning of a cc-nbs-lrr gene from vitis quinquangularis and its expression pattern in response to downy mildew pathogen infection. Mol Genet Genom 293(1):61–68