Article

An Adaptive Epidemiology-Based Approach to Swarm Foraging with Dynamic Deadlines

Hebah ElGibreen ^{1,2,3} ¹ Department of Information Technology, College of Computer and Information Sciences, King Saud University, Riyadh 145111, Saudi Arabia; hjbreen@ksu.edu.sa² Research Affiliate of Mechanical Engineering Department, Massachusetts Institute of Technology, Cambridge, MA 02139, USA³ Member of Center of Smart Robotics Research, College of Computer and Information Science, King Saud University, Riyadh 145111, Saudi Arabia

Abstract: Swarm robotics is an emerging field that can offer efficient solutions to real-world problems with minimal cost. Despite recent developments in the field, however, it is still not sufficiently mature, and challenges clearly remain. The dynamic deadline problem is neglected in the literature, and thus, time-sensitive foraging tasks are still an open research problem. This paper proposes a novel approach—ED_Foraging—that allows simple robots with limited sensing and communication abilities to perform complex foraging tasks that are dynamic and time constrained. A new mathematical model is developed in this paper to utilize epidemiological modeling and predict the dynamics of resource deadlines. Moreover, an improved dynamic task allocation (DTA) method is proposed to assign robots to the most critical region, where a deadline is represented by a state and time. The main goal is to reduce the number of expired resources and collect them as quickly as possible by giving priority to those that are more likely to expire if not collected. The deadlines are unknown and change dynamically. Thus, the robots continuously collect local information throughout their journeys and allocate themselves dynamically to the predicted hotspots. In the experiments, the proposed approach is adapted to four DTA methods and tested with different setups using simulated foot-bot robots. The flexibility, scalability, and robustness of this approach are measured in terms of the foraging and expiration rates. The empirical results support the hypothesis that epidemiological modeling can be utilized to handle foraging tasks that are constrained by dynamic deadlines. It is also confirmed that the proposed DTA method improves the results, which were found to be flexible, scalable, and robust to changes in the number of robots and the map size.

Citation: ElGibreen, H. An Adaptive Epidemiology-Based Approach to Swarm Foraging with Dynamic Deadlines. *Appl. Sci.* **2021**, *11*, 4627. <https://doi.org/10.3390/app11104627>

Academic Editor: Manuel Armada

Received: 31 March 2021

Accepted: 17 May 2021

Published: 19 May 2021

Keywords: foraging; dynamic task allocation; swarm robotics; search-and-rescue mission

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Swarm robotics is an area of robotics that offers efficient solutions to complex real-world problems with minimal cost. One of the most popular swarm robotics tasks is swarm foraging, where swarms can be modeled for different foraging tasks, such as intrusion detection, search and rescue, the collection of hazardous materials, agricultural harvesting, and space exploration [1]. In swarm foraging, the robots collaboratively explore an environment and collect resources for transfer to a collection zone. This task was inspired by adaptive behavior observed in nature, as in insect colonies or bacteria. Foraging using a swarm of simple robots is important for improving performance while efficiently distributing the workload with a low cost [2]. It achieves effective task allocation by interacting with the environment to collaboratively perform complex tasks that are beyond the capability of one robot [3]. The robots should perform dynamic task allocation (DTA) to autonomously determine the search region. DTA partitions the robots into subgroups so that they can collaborate and perform the task together as quickly as possible.

Despite recent advances in swarm robotics, this field is not yet mature, and challenges remain, as is evident from a recent literature review [4,5]. In this paper, we focus on two main issues: modeling and DTA. In regard to modeling, swarm foraging has not matured, due mainly to the absence of generic robots and a universal methodology that can model tasks and be used in many, if not all, applications [4]. There is a need for standardization and simplification, and it is important to propose an adaptable approach that is applicable to real-life problems and can be performed by simple robots that require minimal sensors and controllers. Scalability is a significant challenge in this task because increasing the number of robots can increase the number of inter-robot collisions, while increasing the search area would increase the travel cost. Thus, new scalable and consistent approaches are needed for swarm foraging.

In addition, DTA is another issue that relates to the strategy of dynamically assigning robots to different subtasks to solve the main complex task [6,7]. In foraging, robots have limited communication and sensing ranges, which increases the difficulty of DTA. In recent robotics literature, DTA problems have gained a significant amount of attention [3,8], yet DTA in swarm foraging is still an open research area [9]. Although some work has been conducted on foraging with deadlines, the solutions are still limited [10]. Few studies have analyzed how DTA in swarm foraging fits real-time tasks that must be executed before a deadline. Moreover, currently, DTA methods for dynamic deadlines in swarm foraging do not fully exist, to the authors' knowledge.

Ultimately, to date, there appears to have been limited investigations of swarm foraging for multiple targets including practical aspects of implementation, such as dynamic deadlines [11]. In real life, however, resources have a limited time frame, and this frame can change due to different parameters and the effects of the environment. Thus, a complex scenario with dynamic deadlines is one aspect that needs to be considered in swarm foraging. A formal mathematical model should be proposed, and its DTA should be tested with different aspects of the environment. Therefore, this paper contributes to the literature on swarm foraging as follows:

First, a new epidemiology-based approach is proposed for swarm foraging with online deadlines, called epidemiological foraging with dynamic deadlines (ED_Foraging). In ED_Foraging, a new mathematical model called the dynamic deadline model (DD_Model) is developed for this approach, inspired by the susceptible-exposed-infected-recovered-dead (SEIRD) model [12,13], to predict the expiry rate of resources depending on the information collected by each robot.

Second, a self-organized DTA called hotspot is proposed to utilize the mathematical model developed in the first contribution and allocate the tasks dynamically during foraging. This method allows the robots to allocate themselves to a region without needing continuous communication with other robots. The robots do not have prior knowledge about the resources' real distribution or their deadline states. They search for resources and communicate possible hotspots to the nearest collection zone. The reported information is shared with nearby robots to update their models and predict critical regions. The proposed DTA gives priority to more critical spots to reduce the expiry rate as much as possible.

Third, to analyze the performance of ED_Foraging, the dynamic foraging environment was implemented using the autonomous robots go swarming (ARGoS) simulator [14]. This tool is used to implement the new foraging model with simulated foot-bot robots. The implemented model is then used with different DTAs, configurations, and setups. ED_Foraging is adopted for well-known DTA methods in swarm foraging, i.e., ant colony optimization (ACO), blind, and centralized DTAs. The performance of these methods is compared with that of the hotspot DTA method, which allocates tasks based on the proposed model. The foraging and expiry rate measures are collected from 40 trials repeated with three main design issues. Such issues include flexibility over different setups of an unknown environment, scalability for different numbers of robots, and the robots' robustness on different map sizes. The average and detailed performance is assessed in

addition to the distribution using the Friedman statistical significance test and visualized using box plots.

From practical experiments, it is possible to confirm that the result of using one collection zone is significantly worse than that of using multiple collection zones in terms of the expiration and foraging rates. Moreover, when comparing the DTA methods that use multiple collection zones, it is found that they have similar foraging rates but different expiry rates. From the flexibility experiments, it is concluded that hotspot DTA is best suited to the ED_Foraging approach. It produces consistent results for different map sizes, and it has an expiry rate that is either better than those of the other methods or similar to the best one in most setups. The same result is found when testing the scalability of the methods; that is, the results for the hotspot DTA method are consistent when the number of robots changes, while the other methods, especially ACO and centralized DTA, are highly inconsistent. Finally, in regard to robustness, it can be concluded that the hotspot DTA method is the least affected method. It aims to maintain an average behavior irrespective of the map size even though the resources are reduced.

The remainder of the paper is organized as follows: The related background is discussed in Section 2, and the related works are reviewed in Section 3. The proposed approach is described in detail in Section 4. The experimental results are discussed in Section 5, and the findings are discussed in Section 6. The conclusions and suggestions for future work are presented in Section 7.

2. Background

2.1. Epidemiological Modeling

Epidemiological modeling plays an important role in shaping epidemic dynamics [13]. It can be used to model the transmission probability and, hence, predict the epidemic size, the chains of transmission throughout the population, and the likelihood of epidemic recurrence. Machine learning methods have been used to predict infection rates; however, these methods have not yet been successful in predicting infection hotspots and highlighting critical regions [15]. Statistical epidemic models, on the other hand, have been proposed and validated to model the spread of infectious diseases. Epidemic models, such as the SEIRD model [12], are well-established epidemiological modeling methods that can be used to understand the collected data and predict critical regions with different variables and dynamic environments.

The SEIRD model implicitly assumes mass interaction for an infinitely sized population such that the resources hosting the disease are mixed in the population and the members of the population have identical contact rates. In the SEIRD model, the population is split into five groups: susceptible, exposed, infected, recovered, and dead. In the model, $S(t)$ corresponds to the number of susceptible individuals, who are initialized at time t but can become infected if they come in contact with an infected individual. $E(t)$ corresponds to the number of exposed individuals, which are those who are infected but are not infectious yet. $I(t)$ corresponds to the number of infected individuals, who have the disease at time t and may infect susceptible individuals. $R(t)$ corresponds to the number of recovered individuals. Finally, $D(t)$ corresponds to the number of dead individuals. The sum of all the groups should be equal to the population size, i.e., $S + E + I + R + D = P$, where P is the total number of individuals in the population.

In this model, natural death cases and births are neglected. Additionally, once an individual has recovered, it is assumed that they will be quarantined and, hence, will not be susceptible again. To predict the spread rate in a population, a parameter called R_0 is used as an important indicator. R_0 corresponds to the average reproduction number, which is the number of secondary infected individuals, who were infected by primary infected individuals [16]. R_0 is a key quantity that predicts the evolution of an epidemic. Epidemiologically, a lower R_0 is desirable because it means that the infection growth rate is smaller.

2.2. Swarm Foraging

Swarm robotics is a field of robotics in which a collection of robots collaborate with each other to complete a task [17]. It draws inspiration from animal societies and insect colonies, which have several advantages, such as flexibility, scalability, and fault tolerance [18]. The main characteristic of robots in this field is to be simple and resist communication failures; thus, they are more scalable than one complex robot. Hence, the cost of the robots is relatively low, and they can reallocate themselves.

One of the most popular swarm intelligent tasks is foraging. Foraging behavior can be observed in insect colonies such as ants and bees. In these colonies, the insects behave intelligently to select the richest source of food or the shortest path to a food source [18]. The foraging task can be divided into different subtasks, including exploring the environment, searching for food (resources), transporting food to a safe area called the nest, and navigating the environment while bringing back food. Hence, labor division (i.e., DTA) is necessary in foraging to allocate the robots and collect as many resources as possible.

DTA in swarm foraging involves dividing the task into interdependent subtasks and assigning the robots to perform them in parallel [3]. Since the main task in swarm foraging is to collect resources, the subtasks are regions of the foraging area, which are divided among the robots so that they can collect resources simultaneously. To improve the overall performance in the foraging task, the allocation should be dynamic and adaptive. DTA in foraging can be divided into three types: centralized, negotiation-based, and self-organized [18]. Centralized DTA requires a centralized coordinator that communicates with all the robots and allocates the tasks depending on the information it collects. In this type of allocation, availability and robustness are major concerns, and any communication issue with the central coordinator can cause a failure in the whole system, even when all the robots are working properly. Negotiation-based DTA allows all the robots to continuously communicate and negotiate which tasks to do. Even though this type of task allocation is very robust, it requires a high rate of communication among the robots, which can be difficult to achieve when the rescue area is sparse and the connections are limited. Finally, self-organized DTA allows each robot to make its decision autonomously based on the information it collects while having limited communication with its neighboring peers. Self-organized DTA is observed in nature, where peers autonomously make decisions with limited communication and no centralized control. This type of DTA is less prone to catastrophic failures than other types and is a better approach when the environment is changing rapidly.

3. Literature Review

DTA for swarm robotics has gained much attention in the last few years. In [10], a self-organized task allocation method was proposed to partition a task among multiple robots in a swarm with limited information about the environment. The proposed method does not require explicit communication among the robots. That is, each robot determines its foraging strategy and which subtask to perform based on its local estimate of the global execution time, which is updated after each task is completed. This method was tested in an ARGoS simulator, and it was found that the robots' decision-making improved, which led to overall improvements in performance. However, even though pheromone-based memory can be advantageous in foraging, it performs task allocation under ideal conditions, i.e., when all tasks are equally important and the robots have equal capabilities. Moreover, there can be a tradeoff between the improvement in performance and the scalability of the method. Chung et al. [19] also addressed the self-organized task allocation problem for swarms. They proposed a novel game-theoretical autonomous decision-making framework, which was tested in a dynamic environment. The main goal of their proposed framework was to achieve consensus among all agents regarding the best assignment for a current task. Even though the proposed framework was shown to perform well, it did not compare to existing methods.

In [20], four different DTAs in swarm foraging were proposed. The first algorithm, Random-Choice, randomly selects the tasks and runs in a constant time. The second algorithm, Extreme-Comm, reduces the execution time but requires heavy communication among the robots. The third algorithm, Card-Dealer, sequentially assigns tasks to the robots to reduce communication but increases the execution time. The fourth algorithm, Tree-Recolor, combines Card-Dealer and Extreme-Comm to balance the execution time and communication. The algorithms were tested on a group of 25 iRobot SwarmBots. In the case of Random-Choice, communication was not required, but it had a high failure rate with small- to medium-sized swarms. Extreme-Comm was fast and accurate but required intensive communication that was heavily interrupted. Card-Dealer minimized the need for heavy communication, but the execution time was long because of its sequential task assignment. Finally, in the Tree-Recolor algorithm, problems arose due to communication hardware limitations. A two-step scheme for DTA was proposed by Wei et al. [6] in swarm foraging. It first partitions the task and then autonomously allocates the subtasks to the robots. From the empirical test, it was found that the proposed task allocation approach outperformed the conventional evolutionary-based approach and was more effective.

A DTA method was proposed by Lee et al. [3] to perform self-organizing swarm foraging based on a response threshold model to allocate sequentially interdependent subtasks. The proposed model considered the demand of a given task in addition to the response threshold for the task; the demand decreases while the robot performs the task, and a robot with a higher response threshold is less likely to perform the task. This model regulates the proportion of robots with respect to the fraction of the task demand. Another threshold-based self-organized DTA method for swarm foraging was proposed in [21], to autonomously change the number of working robots. A sigmoid model was used to compute the response threshold, and then each robot could decide whether it should forage to reduce the traffic and congestion in a region. Lee and Kim [8] proposed a decentralized strategy to improve the response threshold model for DTA in swarm foraging. A task selection probability function was proposed for each robot to balance the task demands with respect to the working robots. Although the environment was unknown, each robot could use its local information in addition to the number of tasks assigned to its neighbors. The robots determined their tasks based on their collected information even if the environment changed. From the empirical study, it was found that the proposed system converged to a certain task distribution to reduce the number of changes in the tasks. However, to maintain this performance, the division of labor among the robots needed global information about the environment. This was overcome by introducing two history queues for each robot, which increased the memory demand.

Nature-inspired DTA methods have also been adopted in swarm foraging. In [22], a particle swarm optimization (PSO) DTA method was proposed for swarm foraging. In this method, decisions are made independently by each robot without the need for continuous communication to avoid connection limitations. The simulation results showed that the proposed algorithm could adapt to changes in the environment. However, it is not applicable when the swarm size is large because it compromises between time and optimality. An improved PSO-based approach called potential field-based PSO (IPPSO) was proposed in [23] to perform swarm foraging in an unknown environment. IPPSO was designed to save computation time; its applicability was empirically evaluated in various scenarios to show that the foraging system could complete a task in an unknown environment.

Jevtic et al. [24] proposed the distributed bees algorithm (DBA), which was inspired by the foraging behavior of bees, where there is decentralized decision-making to achieve a global objective. The algorithm was performed with three physical robots, and a simulation was carried out to test its scalability over a large number of swarms. The experimental results showed that the DBA was scalable to the number of targets and robots while being adaptable to the targets' distribution. However, the optimization parameters need to be continuously controlled to adjust the robot behavior, which is difficult to generalize and requires tuning.

Another nature-inspired algorithm is ACO, proposed by Lu [1], which aims to be scalable and efficient for tens to thousands of robots. Lu proposed two algorithms: the multiple-place foraging algorithm (MPFA) and the central-place foraging algorithm (CPFA) [25]. The MPFA was further improved by introducing dynamic depots as special robots that could carry multiple resources [26]. Another extended version based on bioinspired hierarchical branching was published in [27]. To ensure the delivery of resources to the collection zones, the natural phenomenon of delivering blood from the heart to isolated cells in the cardiovascular system was adopted. Through Lu's work, the performance of the MPFA was compared with that of the CPFA on physical and ARGoS simulated robots. It was confirmed that the MPFA produced better foraging rates and travel/search times as well as a lower number of collisions compared to the CPFA. However, neither methods are sufficiently scalable, and they suffer from high collision rates. Moreover, ACO requires continuous parameter tuning, which trades off the search time with the traveling distance.

An ACO-based collective foraging method was proposed in [7], where robots coordinated in an unknown environment to collect resources. The robots simply used binary sensors with a minimal set of states to follow a reactive behavior mode. The proposed method managed to approximate the optimal allocation based on a quality-sensitive modulation of the pheromones. The swarm robotics proposed was used as a tool in understanding biological systems. Another ACO-based method for swarm foraging was proposed in [28]. A dynamic wave expansion neural network (DWENN) was developed based on a neural network to model pheromone diffusion. Based on mathematical modeling, this method determines the key parameters of foraging, including the number of robots assigned to different tasks. Two empirical tests were conducted and showed that the results agreed with the proposed model. However, the tested cases were simplified to produce an exact model, while foraging behaviors are complex.

Song et al. [28] developed a neural network-based pheromone model for swarm foraging. The main purpose of their proposed method was to determine the key parameters for swarm foraging by updating the neural network output based on the robots' pheromones. The reported results showed that the proposed algorithm may face difficulties in large environments. If there are obstacles, the robot might stay in the obstacle-avoidance mode for a long period of time, which means that it is not scalable or adaptable to large or dynamic environments, and further improvements are needed. Nevertheless, bacteria-like behavior is another nature-inspired concept that has been adapted to DTA in swarm foraging. Kurdi et al. [29] proposed a bacteria-inspired heuristic for DTA in unmanned aerial vehicles (UAVs). The proposed approach was tested with three different heuristics and was found to maintain a steady execution time in different environmental contexts. However, the proposed method is problem-specific, and the tasks were assumed to be independent.

The literature discussed above assumes that tasks have equal priority, and deadlines are not considered. However, deadlines are sometimes incorporated into swarm foraging to reflect real-life problems. In this case, the performance of the system is related not only to how fast it can collect food but also to how much food is delivered on time. Thus, methods of swarm foraging have been improved to include temporal constraints. Khaluf et al. [30] developed a new swarm foraging task with soft deadlines and proposed a mathematical model that can be used to analyze the performance of swarm foraging. Their proposed model allows robots to select their tasks autonomously using a decision matrix, which defines the probability of robot behavior. This probabilistic design was tested in an ARGoS simulator and was found to be efficient when the cost of switching between tasks is neglected. Wei et al. [2] further studied temporal constraints with swarm foraging. In their study, the collected resources were categorized into different types. Although the robots have no prior knowledge about the locations of the targets in the environment, they need to deliver the targets in chronological order depending on their types. A prediction auction-based approach was also proposed, where the robot allocates itself while it executes its tasks. The robot predicts the tasks that need to be performed in the future when it is idle. To test the performance of the proposed approach, the blocks world for teams

(BW4T) simulator was used to determine that there was a trade-off between the energy consumption and execution time.

An ACO-based algorithm was proposed by Vanhee [17] to distribute robots into subtasks and finish before a certain deadline. The proposed algorithm was tested on an ARGoS simulator to determine that ACO can be used for foraging tasks with deadline constraints. However, continuous parameter tuning is required to successfully complete the task. Khaluf and Rammig [31] proposed a self-organized DTA strategy to assign time-constrained tasks to swarm robots. A probability allocation matrix was proposed with respect to different specifications, including the robot performance, soft task deadline, and task size. The main goal was to allow each robot to assign itself to a task depending on its deadline. From the ARGoS simulator results, it was found that even though the proposed strategy could adapt to online arrival tasks, it required neglecting the costs of dropping one task to switch to another. Another ACO algorithm was proposed by Khaluf et al. [9] to efficiently assign time-constrained tasks to swarm robots. The proposed algorithm allows the robots to communicate with other robots and periodically evaluate the quality of the allocation using pheromone trails to reduce the execution time. It was tested in static and dynamic environments to show that it performed well with a small number of robots, and parameter tuning was required for a dynamic environment where the optimal values were unknown.

Chung et al. [19] surveyed the literature on aerial swarms and argued that there is a need to improve the methods currently used in swarm robotics. There is an important trade-off between robustness, efficiency, and performance in the current literature. Moreover, this study revealed that swarm robotics is still an open problem and that methods need to be improved in different areas, including decision-making in an unknown environment. Ultimately, it can be concluded that studies on swarm foraging with time constraints are still limited. In particular, the problem of dynamic unknown deadlines has not been considered in the current literature, and time constraints are usually configured before starting the foraging task. However, in reality, deadlines are continuously changing and cannot be predefined. Therefore, further improvements are needed, and dynamic deadlines should be modeled.

4. The Proposed Approach

4.1. System Design

Inspired by MPFA modeling [25], ED_Foraging can be defined as a search-and-rescue mission for individuals (resources) with dynamic deadlines. To clarify how the system is designed to allocate the tasks throughout the foraging journey, this section discusses the model proposed to represent dynamic deadlines and then presents the problem definition to show how the model is designed for the foraging task.

4.1.1. Dynamic Deadline Model

In the past few months, due to the global pandemic of the novel coronavirus disease 2019 [16], epidemiological modeling has been evolving quickly to enable researchers to study virus transmission and model it as accurately as possible. The way the virus behaves and dynamically affects different people is very interesting. Thus, different aspects of this epidemic model can be used to generalize the adaptation and change over time to construct a mathematical model for swarm foraging to predict unknown dynamic deadlines.

To understand how the proposed mathematical model is adapted from epidemiological modeling, it is important to first identify similarities between the two concepts. In swarm foraging, robots move resources into collection zones before they expire. When the deadlines are dynamic, the robot must know its transmission rate to predict the likelihood of deadline expiration in a certain region. Epidemiological modeling can be utilized to predict such transmission, where predicting possible exposure in a population to help victims before they die is an analogy for predicting deadline expiration in an area to move

resources into collection zones before they expire. Thus, infected resources are an analogy for constrained resources.

As shown in Table 1, ED_Foraging performs its task in a 2D arena (A). The size of this arena represents the epidemic size in epidemiological modeling. Multiple robots (R) are used to collaboratively accomplish the task, which involves moving resources from one place to another. The resources should be moved to a depot collection zone (N_d) temporarily until they are moved to the central collection zone (N_c). The resources (N_g) in ED_Foraging represent the individuals in an epidemic population. Moreover, the deadlines are represented by T_ζ , where ζ represents the resource state. Hence, the infection state is mapped to a certain time constraint that represents the deadlines and can be updated online using the proposed epidemic model. Note that each infection state is taken as a deadline, and the death state symbolizes deadline expiration. The transmission of an infection throughout the population can be used to update the deadlines dynamically and predict the likelihood of epidemic recurrence to assign the robots to the most critical regions.

Table 1. ED_foraging analogy.

SWARM Foraging	Symbol
Arena	A
Robots	R
Resources (tags)	N_g
Central Collection Zone	N_c
Depot Collection Zones	N_d
Deadlines	T_ζ

Inspired by the SEIRD model [12], a dynamic deadline model called the DD_Model is proposed in this paper to model the dynamics of the time constraints. The DD_Model is used to predict the state of each resource and update their deadlines online during the task. In particular, there are six states that a resource can be in (transmission, susceptible, exposed, infected, recovered, and dead). The following parameters are used in the DD_Model:

- α corresponds to the incubation rate;
- β corresponds to the exposure rate;
- γ corresponds to the recovery rate;
- η corresponds to the mortality rate; and
- ε corresponds to the duration of immunity for a resource after moving to a recovery state.

Each resource has its own deadline, which is computed with respect to the current time and state. A resource state is governed by one of the deadlines below, where T is the current time, T_0 is the initial time before any state starts, and R_{exp} is a uniform random exponential function with a value that ranges from zero to the rate parameter.

Incubation period T_I —the average time for a resource to enter the infected state after the exposed state.

$$T_I = T_0 + R_{exp}(\alpha) * T \quad (1)$$

Transmission time T_t —the average time for a resource to contact other resources, during which it can infect its neighbors, before the recovered state.

$$T_t = T_I + R_{exp}(\beta) * T \quad (2)$$

Recovery time T_r —the time needed to reach the recovered state after entering the infected state.

$$T_r = T_t + R_{exp}(\gamma) * T \quad (3)$$

Susceptibility time T_s —the time needed for a resource to enter the susceptible state after the recovered state without entering the exposed state.

$$T_s = T_r + \varepsilon \quad (4)$$

Expiration time T_d —the time before a resource expires after being in the infected state.

$$T_d = T_I + R_{exp}(\eta) * T \quad (5)$$

Table 2 shows how a resource stays in each deadline state as long as the time of that state has not expired. Note that the deadlines are continuously updated depending on the current time and previous state. A uniform random exponent is used in the equations to generate different deadlines for the resources. This is of particular importance for increasing the dynamics of deadlines where the time taken to transition from one state to another differs between resources.

Table 2. Deadlines in the DD_model.

State	Deadline
Exposed	$T_0 < T < T_I$
Infected	$T_I < T < T_t$
Recovered	$T_t < T < T_r$
Susceptible	$T_r < T < T_s$
Dead	$T > T_d$

Finally, when a resource is in the infected state, it is possible to predict how many other susceptible resources may enter the infected state in the future. This is carried out through the reproduction parameter R_0 , which is computed based on the current population state, as shown in (6), where $E(t)$, $I(t)$, $R(t)$, and $D(t)$ are the numbers of exposed, infected, recovered, and dead resources, respectively.

$$R_0 = \frac{E(t) + I(t)}{R(t) + D(t)} \quad (6)$$

4.1.2. Foraging Task Model

For a swarm of R robots searching for N_g resources that have deadlines T_c , the task can be formulated as follows:

Definition 1. The robots are divided into two types: searching robots (R_s) and foraging robots (R_f). The searching robots are responsible for searching for resources and moving them to the depots. The foraging robots are responsible for delivering resources from the depots to the central collection zone.

Definition 2. The velocity of R_f is constant across all experiments, and R_s can always carry one resource at a time.

Definition 3. The collection zones are divided into two types: a central collection zone N_c and distributed zones N_d (depots). These zones are distributed uniformly and are known to the robots. The collected resources are moved to the closest zone; then, they are moved to the central collection zone. For each search region, there is a specified area A_r with a collection zone in the center. The number of search regions is N_r , so the total arena area is $A = N_r A_r$.

Definition 4. A resource is associated with a list of events (L_t) representing its states, where $L_t \in \{S, E, I, R, D\}$. Each event is constrained by an online deadline (T_c) calculated using the DD_Model to predict future states and their deadlines.

Definition 5. T_ζ is the time representing the deadline of each resource, where $\zeta \in \{I, t, r, s, d\}$. Each deadline is updated continuously using the DD_Model, as explained in Equations (1)–(5).

Definition 6. The site density (D_t) is the density of an area A_r based on the number of resources (N_g) in A_r weighted by their current states, where N_g is distributed uniformly in A_r . D_t can be measured using (7), where $\alpha_{state(i)}$ is the state weight of a resource (i) in proportion to the states in the DD_Model: $\{\alpha_s = 1, \alpha_E = 2, \alpha_I = 3, \alpha_R = 4, \alpha_D = 5\}$.

$$D_t = \sum_{n=1}^{N_g} \prod_{i=N_g}^n \alpha_{state(i)} \quad (7)$$

Definition 7. The spot intensity (H_t), which is also called the hotspot level, is the predicted maximum energy of a spot in an area A_r based on the average time of resource expiration in that spot predicted by the DD_Model. H_t is computed using (8), where $f(n)$ is the expected expiration time T_d of the expected expired events in A_r and $C(d)_{N_g}$ is the number of resources N_g that are predicted to die in A_r .

$$H_t = \frac{\sum_{n=1}^{N_g} f(n)}{C(d)_{N_g}}; \text{ where } f(n) = \begin{cases} T_d; & \text{if } D \in L_t(n) \\ 0; & \text{else} \end{cases} \quad (8)$$

Definition 8. The collected resources are replenished after collection or expiration to maintain a consistent density. When a resource is collected, another resource is placed in a location drawn from a uniform random distribution. Moreover, when a resource expires, another susceptible resource is placed in another location with a new constraint based on the DD_Model.

Definition 9. The foraging rate (F) is the number of resources collected in the central collection zone N_c per unit time. The foraging rate in a region is F_d , which represents the number of resources transported to the regional collection zone N_d .

Definition 10. During the search, the robots use a search probability $\rho \in (0; 1)$ to decide whether they should travel based on the selected locations (H_t or D_t) or switch to a uniform random search at each time step. This probability controls the distance and time that each robot travels in a straight line and allows flexibility in searching the area based on a uniform random distribution.

At the beginning of the task, the resources are randomly distributed around the area, and the DD_Model is used to randomly change the resource states to infected and set their initial deadlines. The robots in this task have limited communication ranges and sensing capabilities, yet they need to work together to collect as many resources as possible before their deadlines. The resource locations are unknown to the robots; hence, they blindly search while collecting information about their surroundings and report the information when delivering resources to the closest zone they find.

All types of robots can move globally around the area but searching robots R_s return to the closest depot when they collect a resource. When a resource is collected by an R_s , the D_t and H_t of its surroundings are computed and reported to the closest collection zone. This information can be used by other robots to give priority to intense spots. Robots that detect a high density of resources D_t in a certain region have a high probability of returning to that region. This probability is defined by a Poisson cumulative distribution function (CDF) $\text{POIS}(D_t; p)$, as shown in (9), where p is the reliability parameter set by the user. The robot returns to its previous location when the parameterized Poisson CDF exceeds a uniform random value, i.e., $\text{POIS}(D_t; p) > \rho \in (0; 1)$; otherwise, the robot returns to one of

the hotspots in its region, if any exist. If there are no hotspots, then the robot switches to a uniform random search.

$$POIS(k; p) = e^{-p} * \sum_{i=0}^{D_t} \frac{p^i}{i!} \quad (9)$$

The main objective of this task is to reduce the number of expired deadlines by observing the environment and updating D_t and H_t accordingly to collect resources as quickly as possible to give priority to hotspots.

4.2. Methods and Algorithms

After formulating the problem at hand, we now show how the ED_Foraging approach works and explain the details of the proposed DTA method, called Hotspot DTA. As illustrated in Figure 1, the approach starts by initializing the state of all resources to susceptible and randomly distributing them around the arena. Then, it randomly changes some of the resources' states from susceptible to exposed via the DD_Model. Finally, robots are dispatched from the collection zones to follow randomly selected paths, and the constraints are updated online at each time.

```

ED_Foraging()
//[1] Initialization
Initialize Foraging environment    //place depots and robots
Initialize DD_Model and set all Ng states to susceptible
//[2] Distribute resources
∀ Ng[i] ∈ Ng
    Randomly distribute Ng[i] based on distribution X~
//[3] Infect part of the population; change the state from S to E
Max_Infection = Ng.size() + 0.75*Ng.size()
For (i = 0 .. Max_Infection):
    Infected_I = randomly get a "susceptible" resource from Ng
    InfectionList += Infect (Infected_I, Current_Time)
//[4] While searching, collect resources, update the deadline, and update the hotspot level
While (CurrentTime < MaxTime) AND (CollectedResource < Ng.size()):
    Robot_Search_and_Rescue()
    Update_DD_Model()
    Update_Hotspot_Intensity()
  
```

Figure 1. ED_Foraging "Main function".

To allow a swarm of robots to work together and collect resources without prior knowledge about their distributions or deadlines, the hotspot DTA method is proposed using the *Robot_Search_and_Rescue* function, which is continuously executed, as shown in Figure 2a. In this function, two important actions are configured. The first action is that when the robot finds a resource, it picks it up and delivers it to the closest collection zone. While the robot is picking up the resource, the robot scans its surroundings to compute its spot intensity H_t and local density D_t . As described in (7) and (8), D_t is measured to allow the robot to remember the location of a previously found target and return to the same location if it is dense, while H_t is computed based on the average expiration time of the expected expired deadlines surrounding the robot to prioritize more critical locations.

Figure 2. ED_Foraging “Hotspot DTA functions”. (a) Important actions performed by the robot during its search; (b) Assigning a robot to a hotspot using its weight, initialized as H_t .

The second action is that when the robot reaches a collection zone, it delivers the resource that it is carrying. In this case, the robot reports the collected information (H_t) to the collection zone, which is shared with the other robots. Then, the robot checks the density D_t using the Poisson CDF, as discussed in the previous section. If the Poisson CDF $< \rho \in (0; 1)$, then the last visited site is chosen, and the robot returns to a pre-visited region with a high density of resources. However, if the Poisson CDF condition is not satisfied, the robot checks whether there are any hotspots around it. Using the *HotspotExist* function, the robot scans the hotspot list stored in the current collection zone to choose intense regions based on H_t (Figure 2). The chosen hotspot is then considered as the navigation target so that the robot goes to and searches for resources in that area. Note that the spot intensity (weight) is initialized by H_t via (8) and updated continuously by all the robots using the *Update_Hotspot_Intensity* function; this function will be discussed later in this section.

Going back to Figure 2a, if there is no intense hotspot or dense region found, the robot randomly searches any region. During the random walk, if the searching time exceeds the search probability, the robot gives up searching that region and randomly searches in another region. While the algorithm is executing the *Robot_Search_and_Rescue* function, it also updates the *DD_Model* and the deadlines accordingly. From Figure 3a, the *Update_DD_Model* function is used to continuously check the constraints and update the resource state.

Figure 3. ED_Foraging “DD_Model functions”. (a) The resource states are validated and updated according to the DD_Model. (b) The expected time constraint of each resource is computed and added to the resource event list L_t . (c) The resource constraint is updated depending on the resource’s current state and expected future events stored in the event list L_t .

In the beginning, the *Update_DD_Model* function checks whether there is a deadline at the current time. If no active deadline is found, the model randomly adds deadlines by changing the state of the susceptible resources using the *infect* function to introduce new deadlines to the task. On the other hand, if there are resources with deadlines, the earliest deadline state is checked. If the deadline is greater than the current time, the resource state is updated to the next state, and hence, the deadline becomes more critical. The state of the earliest resource is updated depending on its current state and the deadline via the *update_constraint* function. After updating the deadline, the new resource state is validated. If the new state is dead, it means that T_d has been reached and the resource deadline has expired. Thus, it should be removed from the population and reassigned to a new location as a new susceptible resource.

In the *infect* function (Figure 3b), the expected deadlines of the selected resources are computed with respect to the current time using (1)–(5). Then, the current state of the infected resource is set as exposed. To avoid computation overload, the expected deadline and future state of an infected resource are stored in a queue list called *InfectionList*. This list represents the event list (L_t) and contains all expected events in the population, arranged in order with respect to their deadlines as shown in Table 2. Thus, the states following exposure are added for the current resources, including infection, recovery, transmission, susceptibility, and death. In each state, the name of the state is added along with its deadline and the expected resource index.

In the *update_constraint* function (Figure 3c), when the state is infection, recovery, susceptibility, or death, the subsequent state of the chosen resource is updated based on L_t as computed by the *infect* function (Figure 3b). If the new state is transmission, this means that the resource might change its neighboring resource states to infection until its own state is changed to recovery. Thus, based on the reproduction parameter R_0 computed in (6), random neighbors of the current resource are selected, and their state deadlines and event lists are updated via the *infect* function (Figure 3b).

Throughout the ED_Foraging task, the robots continuously search for resources, giving priority to dense and intense spots. During the search, the deadlines are dynamic, and thus, the *Update_Hotspot_Intensity* function (Figure 4) is used to continuously update the hotspots based on the current infection state of each region. This function updates the intensity of the recorded spots added in the *Robot_Search_and_Rescue* function (Figure 2a). The intensity of the spots predicted to have expired resources is updated by subtracting the current time from the predicted expiration time (computed using (8)). However, when a spot is predicted not to expire, which applies to spots that have noncritical resources, the intensity is reduced exponentially with respect to the current time, where y is the decay probability set by the user.

```

Update_Hotspot_Intensity()
//[1] For every hotspot, update its intensity weight
∀ Hi ∈ Nest · Hotspot:
    If Hi is predicted to have at least one dead case
        Hi.weight = Hi.Ht - Current_Time //Ht was computed using eq(8)
    Else//intensity decays exponentially with time
        Hi.weight = Hi.weight *  $e^{-y \cdot (current\_time - t)}$ 
        t = current_Time

//[2] If this spot is not a hotspot, remove it from Ht
If Hi.weight ≈ 0 THEN
    Nest.Hotspot = Nest.Hotspot - Hi

```

Figure 4. ED_Foraging “Hotspot Intensity function”.

After updating the intensities of all hotspots, if the weight of a spot is almost equal to zero, then it is no longer considered a hotspot and is predicted to have no critical resources at the current time. Thus, it should be removed from the hotspot list. Throughout the task, the robots continuously search for resources, giving priority to hotspots with infected resources that could die if not rescued. During the search, the deadlines are dynamic and are changed online, and the DD_Model is used by the robots to predict important regions.

5. Experiment

To measure the performance of the proposed ED_Foraging approach, it was implemented in an ARGoS simulator [14,32] with foot-bot robots. As illustrated in Figure 5, ARGoS is a 3D open-source software for real-life simulations of multi-robot swarm tasks. In this section, the detailed setup and configuration of the executed experiments are presented. Then, the results of the different tests are discussed. Finally, the important findings are summarized at the end of the section.

Figure 5. Screenshot of multi-place foraging in the ARGoS simulator: Bird-eye vs. Zoomed screenshot.

5.1. Experimental Components

The experiments were run on an i7 machine with 16 GB RAM. Each experiment was repeated 40 times to record the average and specific performances. The maximum simulation time of each trial was 30 min. The rate parameters in the DD_Model were initialized based on the Hladish et al. [13] study, where $\alpha = 0.5$, $\beta = 0.33$, $\gamma = 0.17$, and $\varepsilon = 30$. It was assumed in the experiments that any resource could have a deadline and expire if not collected before its expiration time. Thus, the mortality rate was initialized as $\eta = 1 - \gamma$.

To study how ED_Foraging can manage online deadlines in an unknown environment, the proposed foraging approach was tested with different DTA methods, including hotspot DTA. The first three methods are self-organized DTA, and the fourth method is centralized DTA.

- Hotspot—The proposed DTA method discussed in Section 4.
- ACO—Adopted from the MPFA, which is ACO-based DTA with multiple collection zones, where ACO parameters are initialized as in [25].
- Blind—Implemented to allocate robots randomly and search blindly in multiple collection zones. No information is reported on the zones or exchanged between the robots.
- Centralized—A traditional DTA for swarm foraging, where there is only one collection zone (central) and the search is blind and random.

Throughout the experiments, different measures were collected. The first measure was the foraging rate, which represents the number of resources delivered to the central collection zone (a higher foraging rate corresponds to a better performance). The second measure was the expiry rate, which represents the percentage of resources with expired deadlines. A lower expiry rate corresponds to a better performance. The expiry rate was computed by dividing the number of resources that died (expired deadlines) by the total number of resources scheduled (expected) to die.

5.2. Experimental Design

To study the performance of ED_Foraging, three experiments were performed to assess its flexibility, scalability, and robustness. As shown in Table 3, the first experiment measured the flexibility of the approach with different setups, where the number of robots and resources and collection zones changed with respect to the map size. Each experiment was tested with different configurations, from small to very large maps and numbers of robots, depots, and resources.

Table 3. Experimental configurations.

Experiment	Map Size	R	N_g	N_d
Flexibility	4×4	24	16	5
	8×8	112	64	17
	16×16	136	256	21
	20×20	144	400	21
	40×40	592	1600	85
Scalability	16×16	24	256	21
		112		
		136		
		144		
		592		
Robustness	4×4	112	64	5
	8×8			17
	16×16			21
	20×20			21
	40×40			85

The second experiment, however, measures the scalability of the approach by changing the number of robots and fixing the rest of the configuration. The map size was fixed at 16×16 because DTA methods using multiple collection zones (i.e., ACO, hotspot, and blind) ended the experiments before moving the resources to the central zone. This means that, in this map, the robots were scalable enough to finish collecting all the resources before moving them to the central zone. Thus, it was important to study this behavior and further investigate the scalability of each method. Note that since resources were not moved to the central zone in the 16×16 map, the foraging rate represents the total number of resources collected in all zones, i.e., $\sum_{\forall N_d} F_d$.

Finally, the third experiment measured the robustness of the approach by changing the map size, and accordingly the collection zones, and fixing the number of robots. Note that the centralized method uses only one collection zone, regardless of the map size.

5.3. Experimental Results

As mentioned previously, two measures were quantified in each experiment and three experiments with different setups were conducted. In this section, the average results of the foraging and expiry rates for each experiment are discussed. Then, the expiry rate is further investigated by presenting the detailed results of the 40 trials. Finally, based on the Friedman statistical significance test, the distribution of the expiry rate is visualized in box plots to discuss how each method behaves in each setup.

5.3.1. Flexibility

To analyze the flexibility of ED_Foraging with different setups, the average performance over 40 trials was collected for both the foraging and expiry rates (Figure 6). From Figure 6a, it is clear that the DTA methods that use multiple collection zones, i.e., hotspot, ACO, and blind, have almost the same foraging rates, while centralized DTA produces the lowest foraging rate in most maps. These results are in agreement with those of Khaluf [18], where self-organized DTA is better when the environment changes rapidly. Moreover, in regard to the expiry rate (Figure 6b), centralized DTA also produces the worst average expiry rate in all cases, indicating that it is better to have multiple collection zones to treat patients before they die. Nevertheless, ACO, hotspot allocation, and blind allocation behave differently, and the expiry rate results need to be further investigated to compare their performance in each map. However, it can be concluded that hotspot DTA produces the best average expiry rate in most maps when applied to ED_Foraging, followed by ACO and blind DTA.

Figure 6. Average flexibility of the different methods in terms of the (a) foraging and (b) expiry rates.

To further investigate the expiry rate in each map and clarify the differences between the methods, the detailed results of each trial are shown in Figure 7a–e. In all the maps, it is clear that the centralized method produces significantly worse results than the other methods. In the 4×4 and 8×8 maps (Figure 7a,b), ACO, hotspot, and blind DTA perform very similarly in all the trials and produce almost the same expiry rate, although the hotspot method produces slightly lower values. However, when the map size is increased, the differences become more apparent.

In the 16×16 map (Figure 7c), the hotspot DTA has the lowest expiry rate in most trials, followed by the ACO and blind methods. In the 20×20 map (Figure 7d), the results for the hotspot and ACO methods coincide for the first half of the trials, indicating that they perform similarly, while the blind method is worse than both. However, for the next half of the trials, the hotspot and blind methods are similar, while the ACO method is worse. Finally, in the 40×40 map (Figure 7e), the hotspot and ACO methods perform similarly for the first half of the trials, while the blind method is worse than both. Then, all three methods have similar expiry rates for the next half of the trials.

To further investigate the distribution of the results in each map and see how hotspot DTA performs compared to the other methods, the distributions of the results are shown in Figure 7f as box plots. In all the maps, it is clear that centralized allocation performs the worst, as previously found. The result of this method is highly variable, indicating its inconsistency between trials. Moreover, the minimum-maximum-mean lines in the plot show that centralized allocation produces a significantly higher expiry rate than the other methods. In regard to the ACO, hotspot, and blind methods, the distributions (Figure 7f) are different between the maps depending on their sizes. In the 4×4 map, all three methods have similar minimum-maximum-mean values, but the results for the hotspot method are more compressed than those for the ACO and blind methods. This

indicates that the hotspot method is more consistent with small maps when comparing the expiry rate.

— ACO — Hotspot — Blind — Centralized

(a) Expiry rate for the 4×4 map

(b) Expiry rate for the 8×8 map

(c) Expiry rate for the 16×16 map

(d) Expiry rate for the 20×20 map

(e) Expiry rate for the 40×40 map

(f) Distribution of the results for the expiry rates of all maps

Figure 7. Flexibility results for 40 trials with different setups. The results are ordered from smallest to largest for better representation (a–f).

In the 8×8 map, the ACO, hotspot, and blind methods have similar mean values, but the hotspot method has the lowest minimum value; it also has the lowest maximum value, along with the blind method. The ACO method produces slightly worse minimum–maximum results while having a similar dispersion as the hotspot method. On the other hand, the expiry rate for the blind method is more compressed but has more outliers that could affect the results in the long run.

In the 16×16 map, it is clear that the results of the hotspot method are significantly better than those of the other methods. It has lower minimum–maximum–mean expiry rates and is also highly compressed compared to the other methods. This indicates that the hotspot method performs better than the other methods and can better handle the increase

in the map setup. In regard to the 20×20 map, the hotspot method is similar to the ACO and blind methods with a slightly lower expiry rate. Unlike the centralized method, the ACO, hotspot, and blind methods are positively skewed, which means that most of the expiry rate results are on the lower end of the scale (shown as the mean line on the bottom of the box). Moreover, the results of blind DTA are more compressed than those of the other methods in this setup.

Finally, in regard to the 40×40 map, although the ACO, hotspot, and blind methods have similar dispersions, it is very clear that the results of the hotspot method are positively skewed, while those of the ACO and blind methods are negatively skewed. This indicates that most of the expiry rate results of the hotspot method are on the lower end of the scale, and the results of the ACO and blind methods are on the upper end of the scale.

From the results of all the trials discussed, it is clear that hotspot DTA gives results that are either better than or similar to those of other benchmarks tested with ED_Foraging. Moreover, it has better performance than the centralized method and is more consistent than the ACO and blind methods, regardless of the map setup. The methods using multiple collection zones behave similarly in small map setups, but the differences become more apparent as the map size is increased. Therefore, the next sections will further investigate the scalability of the methods and their robustness to change.

5.3.2. Scalability

This section analyzes to what extent the algorithm is scalable with swarms that range from tens to thousands. This is of particular importance because increasing the number of robots produces more inter-robot collisions, which affects the scalability of the algorithms. Thus, adding more robots does not necessarily improve performance, and the algorithm's behavior in such setups needs to be further investigated. Starting from the total foraging rate of all zones, i.e., $\sum_{v \in d} (F_d)$ (see Figure 8a), it can be seen that having a very small or very large number of robots in a medium-sized map is not a good choice and negatively affects the foraging rate. From the figure, the best result is obtained when the number of robots is intermediate (136) with a 16×16 map. This confirms the conclusion in the literature, where it was emphasized that increasing the number of robots can negatively affect the foraging rate and that it is better to set the number of robots in proportion to the map size.

Figure 8. Average scalability of the different methods in terms of the (a) foraging and (b) expiry rates.

One interesting observation from Figure 8a is that the foraging rate is affected more by the zone's setup than by the allocation method used. In particular, all multiple-collection-zone methods (ACO, hotspot, and blind) have similar foraging rates that are better than that of the centralized method, which has only one collection zone. Only in the case when the number of robots is very small (24 robots) does the centralized method produce slightly better results than the other methods. This is because having a very small number of robots reduces the collision and hence improves the foraging rate. Therefore, when increasing the number of robots around one central collection zone (i.e., 112, 136, 144, and 592 robots) in

Figure 8a, the foraging rate is significantly affected in the centralized method, to the point that the results are less than half of what the other methods are able to collect.

In regard to the average expiry rate (Figure 8b), each algorithm behaves differently from one test to another. Thus, the detailed results of each trial will be investigated next. However, in general, it can be noted that the centralized method is highly inconsistent, as it significantly reduces the average expiry rate when the number of robots is small while significantly increasing it when the number of robots is large. This indicates that having one collection zone is not a good choice, especially when we are targeting sensitive search and rescue missions.

Going deeper into the detailed results of the expiry rate (Figure 9a–e), it can be clearly seen that the centralized method is inconsistent. Its expiry rate starts from a very low level (a zero expiry rate) when the number of robots is 24 (Figure 9a) and rises as the number of robots increases (Figure 9b–d) until it becomes significantly higher than those of the other methods, with an 80% expiry rate when the number of robots is 592 (Figure 9e).

In regard to the ACO method, inconsistent behavior can also be observed, similar to that of the centralized method but not as obvious. When the number of robots is very small to medium (Figure 9a–c), the ACO method produces a similar expiry rate to those of the hotspot and blind methods in more than two-thirds of the trials while producing lower expiry rates in one-third of the trials. When the number of robots is large to very large (Figure 9d,e), however, the ACO method behaves in exactly the opposite fashion, producing expiry rates similar to those of the hotspot and blind methods in more than two-thirds of the trials while producing higher expiry rates in one-third of the trials. This means that its results are not consistent, and the expiry rate is greatly affected by the number of robots.

Meanwhile, the expiry rates of the hotspot and blind methods are notably very similar in most cases, and the effect of the number of robots is minimal in comparison to the effect on the other methods. It is observable in all setups (Figure 9d,e) that, regardless of the number of robots used, the expiry rate generally ranges between 20% and 70%. The performance is also not significantly reduced in certain setups but is significantly increased in others. Hence, blindly allocating robots or allocating them based on infection hotspots reduces the effect of inter-robot collisions and utilizes the available resources to achieve a more stable performance than that of the other methods.

From the result distribution (Figure 9f), it is very clear that the centralized method's expiry rate result is highly inconsistent. In the 24-robot experiment, it has a significantly lower minimum-maximum-mean expiry rate value, while in the 592-robot experiment, it notably has the highest minimum-maximum-mean expiry rate value. Similarly, in regard to the ACO method, its results are more dispersed than those of the hotspot and blind methods when the number of robots is 24 and 144; its results are similar to theirs when the number of robots is 112 and 592, and its results are more compressed when the number of robots is 136.

Furthermore, the ACO method results are sometimes positively skewed, while at other times, they are negatively skewed. This indicates that the ACO method is highly inconsistent and is not as scalable as the hotspot and blind methods when the ED_Foraging approach is adopted; the ACO method results in the best outcomes when the number of robots is intermediate (136) and the map size is medium (16×16).

Regarding the hotspot and blind methods, although the detailed results indicate that they have similar outcomes, when their distributions are compared (Figure 9f), the hotspot method results are more compressed than those of the blind method, particularly in the 24-, 136-, and 592-robot experiments. The hotspot method is slightly less compressed than the blind method in the 112-robot experiment, but it is positively skewed, indicating that most of its values are on the lower end of the scale.

Figure 9. Scalability results for 40 trials with different setups. The results are ordered from smallest to largest for better representation (a–f).

5.3.3. Robustness

This section analyzes to what extent the robots are robust to different map sizes when the other components of the environment are held fixed. This is of particular importance because increasing the map size means that the robot travel time will increase and that meeting the deadlines might therefore be more difficult.

Starting with the average foraging rate (Figure 10a), as in the flexibility experiment, the ACO, hotspot, and blind methods have similar foraging rates in all map sizes. The average foraging rate (Figure 10a) of the centralized method, however, is worse than those of the other methods are when the map size is small, medium, and large, although it is better when the map size is very small or very large. This is due to two factors. First, having one collection zone in a very small or very large map when the number of robots is medium can decrease the travel time and hence increase the foraging rate. Second, the foraging rates of the ACO, hotspot, and blind methods in this figure show the average number of

resources found in the central collection zone only (F) and not all collection zones ($\sum F_d$). Thus, the small difference between the foraging rates of the centralized method and the other methods can be attributed to the fact that the other methods allocate robots to move resources to the closest collection zone before they die.

Figure 10. Average robustness of the different methods in terms of the (a) foraging and (b) expiry rates.

In regard to the average expiry rate (Figure 10b), the centralized method has the worst results in all map sizes except the 16×16 map, where it produces the lowest average expiry rate. This behavior will be further investigated when discussing the detailed expiry rates shown in Figure 11. Going back to Figure 10b, in a very small map (4×4) and a very large map (40×40) with a medium number of robots, hotspot and blind DTA are better than the ACO method. In a small map (8×8), however, all three methods perform similarly. In a medium map (16×16), the ACO and hotspot methods have similar robustness, while the blind method performs worse than the rest. In a large map (20×20), the ACO and blind methods perform similarly, while the hotspot method is slightly worse. Therefore, it can be concluded that the hotspot method is either more robust than the other methods or is similar to the best method, except in the 20×20 map.

To further investigate the differences among the methods, especially in the case of the 16×16 and 20×20 maps, the detailed expiry rate of each trial is presented in Figure 11a–e. The centralized method has the worst expiry rate in all trials with the 4×4 map (Figure 11a), 8×8 map (Figure 11b), 20×20 map (Figure 11d), and 40×40 map (Figure 11e). In regard to the 16×16 map (Figure 11c), the centralized method produces very inconsistent results from one trial to another.

As seen from the yellow line in Figure 11c, the expiry rate is significantly lower in half of the trials and significantly higher in the other half. This indicates that even though the average expiry rate of the centralized method is low, this result is not guaranteed, and it is not robust in many cases.

The ACO, hotspot, and blind methods have similar expiry rates in the 8×8 map (Figure 11b), 16×16 map (Figure 11c), and 40×40 map (Figure 11e). However, in the 4×4 map (Figure 11a), the ACO method produces a worse expiry rate than the hotspot and blind methods in most trials. This indicates that ACO is not robust to very small maps with a medium number of robots. Moreover, in the 20×20 map (Figure 11d), the hotspot method produces worse expiry rates than the ACO and blind methods in half of the trials. However, it is interesting that the hotspot method results are more consistent than the results of the other methods. Its results are 40 to 55% in all trials, while the ACO and blind results are 20 to 60%. The ACO and blind methods produce low expiry rates in half of the trials while showing higher expiry rates in other trials. This will be further clarified in the box plot visualization.

Figure 11. Robustness results for 40 trials with different setups. The results are ordered from smallest to largest for better representation (a–f).

Based on the expiry rate distribution shown in the box plot (Figure 11f), it can be confirmed that the centralized method is not robust to changes in the map size. Other than in the 16×16 map, the distribution of the centralized method is the highest and most dispersed, indicating that it has a significantly worse expiry rate than the other methods. In the 16×16 map, the centralized method results extend from 0% to 70% and are highly dispersed. This shows that even though the average expiry rate (Figure 10b) of the centralized method is the lowest in the 16×16 map, from the distribution shown in Figure 11f, this result is not guaranteed, and the expiry rate is highly inconsistent from one trial to another.

In regard to the ACO, hotspot, and blind distributions (Figure 11f), the behavior differs from one map to another. However, the hotspot method behaves consistently in all cases, preserving a similar rank in all map sizes in comparison to the other methods. In the 40×40 map, the ACO method produces the worst expiry rate distribution in comparison to the hotspot and blind methods, and it is highly dispersed and has the maximum values.

The hotspot and blind methods, on the other hand, produce similar expiry rates in this map size. When increasing the size to 8×8 , the blind method has the worst expiry rate distribution, with the hotspot method still ranking in the middle, while the ACO method has the best distribution. In the 16×16 map, however, the blind method produces the best distribution, while the ACO and hotspot methods have similar distributions.

In the 20×20 map, the blind method has the lowest minimum value but the most dispersed result, indicating that its robustness is not consistent. The ACO method is second in regard to the minimum value, but it is negatively skewed, and its results are more dispersed than those of the hotspot method. With the hotspot method, although its average expiry rate (Figure 10b) is slightly higher than those of the ACO and blind methods in the 20×20 map, the distribution shows that its maximum value is lower than that of the other methods and that its results are more compressed than the other methods while also being positively skewed. This indicates that the hotspot method performance in this map is more consistent, i.e., robust, and produces results on the lower end of the scale in comparison to the other methods.

In a very large map (40×40), the ACO, hotspot, and blind methods have the same maximum and mean expiry rates but different minimums and dispersions. The results for the minimum expiry rate and dispersion of the hotspot method fall between the results of the ACO and blind methods. On the other hand, the blind method has the lowest expiry rate value, while ACO has the most compressed results. Therefore, from all the results of the box plot distribution (Figure 11f), it can be concluded that the hotspot method performs consistently, with an intermediate robustness in all map sizes, unlike the other methods, which offer the best results in some maps but the worst results in others.

6. Discussion

ED_Foraging is a novel foraging approach that is proposed to achieve self-organized allocation with dynamic deadlines. A swarm of robots searches an unknown environment to collect and transport constrained resources to a known nest location. The dynamic deadlines of the resources are modeled using the DD_Model, which is developed to predict the state of each resource while the robots search blindly and have no prior knowledge about the resources' locations and states. The proposed hotspot DTA method was able to prioritize critical regions. The intensity of the hotspot regions was adaptively updated, and thus, the allocation was dynamic and depended on the current state of the environment and the DD_Model.

When it comes to the computational complexity of the ED_Foraging approach, two main factors affect the performance: the number of resources N_g and number of hotspots H . The navigation continues as long as there is a resource not collected; i.e., $O(N)$. During this navigation, the approach is to search the hotspots, update the model, and update the hotspots density; i.e., $O(H + N + H)$. Based on the algorithm discussed in Section 4.2, the complexity of the approach can be as follows:

$$\begin{aligned} O(\text{ED_Foraging}) &= O(\text{Navigation}) * [O(\text{hotspot search}) + O(\text{model update}) + O(\text{hotspot update})] \\ &= O(N) * [O(H) + O(N) + O(H)] = O(N) * O(2H + N) \\ &= O(N^2) \end{aligned}$$

This result was concluded because the maximum number of hotspots can become equal to the number of resources but cannot exceed it. Thus, the number of resources needed to be collected affects the approach performance. This means that applying the algorithm on real or simulated data has no effect on the complexity, but it is rather affected by the size of the targeted resources. Hence, the method can be suitable for real robots the same way it was suitable for the simulated one.

When it comes to the experimental results, starting with the flexibility of ED_Foraging, it was confirmed that hotspot DTA produced results that are more consistent. It was significantly better than the rest of the algorithms with a mid-sized map while similar to the best one in other map sizes. The blind and ACO DTA methods had inconsistent expiry

rates since their results dispersed and became worse as the map size increased. Moreover, the ACO method results became negatively skewed with a very large map setup. The hotspot DTA method, however, obtained results that were either better than those of the other methods (in very small-to-small setups) or similar to the best results (for larger setup sizes). This indicates that it adapts better to changes in the environment.

In regards to scalability, the foraging rate was affected more by the number of collection zones than by the DTA method. This is normal because increasing the number of robots while retaining one collection zone naturally increases the number of inter-robot collisions and thus reduces the foraging rate. Nevertheless, this was not the case when comparing the expiry rates. The centralized and ACO DTA methods were highly inconsistent, as they reduced the expiry rate with a small number of robots while increasing it significantly with a large number of robots. They were only suitable in the case where the number of robots was proportional to the map size (an intermediate number of robots and a medium map size). The hotspot and blind DTA methods, on the other hand, were more stable and performed more consistently than the other two methods. In fact, the hotspot DTA method was significantly better than the rest of the algorithms with a large number of robots, while similar to the best in other sizes.

From the robustness experiments, it was found that the hotspot DTA result was usually in the middle compared to the rest of the methods. Meanwhile, centralized DTA was the worst in all cases, while the ACO and blind results were interwoven, such that one was the best in certain cases but the worst in other cases. Hence, the hotspot method tries to maintain a similar performance regardless of the map size, even when the resources are reduced. Its results were consistent and similar to the best one at each case.

Ultimately, although the proposed approach showed a similar foraging rate to the literature, the significance became clearer when comparing the detailed expiry rate. From the results discussed before, it can be concluded that the merits of the proposed approach resides in producing flexible, scalable, and robust results when comparing the expiry rate. Unlike the other methods, in which they might excel in one setup but not the rest, hotspot DTA showed a consistent performance and produced better results in each case, depending on the setup. On the other hand, the ACO method did not perform well with small maps or when there was a large number of robots, the blind method did not perform well with large maps, and the centralized method did not perform well in most cases, even when comparing the foraging rate. This conclusion indicates that the proposed approach can be applied to different real-life application in more than one scenario, as will be discussed next.

Real-Life Applications

The model proposed to predict the states of resources and their deadlines was adopted for epidemiological modeling and, thus, can be mapped to different types of real-life problems in which any epidemiological mathematical model can be adopted. Foraging with online deadlines can also be scaled to a large spectrum of real-life problems, as long as it can be mapped to biological foraging behavior. Khaluf [18] discussed different possible applications of this, such as cleaning pollution that needs to be collected at a specific time, operating factory-recycling systems, or constructing data mules that are used for collecting wireless sensor nodes' stored data. All these applications can also be undertaken using ED_Foraging.

One example of real-life application, in which the proposed approach can be used, is for a COVID-19 search-and-rescue mission. This mission can be about searching for infected individuals and predicting possible exposure around these individuals to collect the victims before they die. Multiple robots can be used to collaboratively accomplish the task, which is about moving objects from one place to another. Collected objects (resources) can represent individuals (or patients), and clinics are represented by the collection zones. In general, the task is about searching for patients, and when there is a suspicion of COVID-19 infection (captured by temperature sensors for example), individuals are moved to checkup points in order to be tested if they are positively infected. When the patients result

is positive, they are moved to a general hospital to receive treatment, which is represented by the central collection zone. Finally, when it comes to the deadlines, it is important to move infected people before they die. Hence, the infection status can be mapped to the time constraint model, which can be updated online using the proposed DD_Model. The ultimate objective is to move as many patients as possible, while considering the time constraint to give priority to individuals that are more critical. Hence, the efficiency of the model is not only concerned with the number of resources it collects (foraging rate) but also about the death probability (expiry rate). Having the foraging and expiry rate are very important metrics to give an intuition about how well the system perform and to what extent the task is accomplished.

From all of the above, it can be concluded that any search-and-rescue mission with online deadlines can also be carried out using the proposed method. All that is required is to decide on the sensors that will be used to identify the resources and update the DD_Model parameters to model the task requirements. The ultimate objective is to prioritize critical individuals rather than to collect a large number of resources.

7. Conclusions

In the ED_Foraging approach, a swarm of robots searches an unknown environment to collect and transport resources with dynamic deadlines to a known nest location. The main contribution of this approach is to address the modeling and DTA problems highlighted in the literature on swarm foraging that have dynamic deadlines. The dynamic deadlines of resources and their states were modeled using the DD_Model, which was developed to predict the state of each resource, although the robots searched blindly and had no prior knowledge about the resources' locations and states. The robots collected resource information and updated the DD_Model when reporting to the closest zone. Then, this shared information was used by nearby robots to continuously update their models and, hence, give priority to denser and more intense regions. Thus, it was possible to deal with dynamic and online deadlines while searching.

In regard to DTA, the proposed hotspot DTA method was able to prioritize critical regions. The intensities of the hotspot regions were continuously updated, and thus, the allocation was dynamic, depending on the current state of the environment and the DD_Model. The main objective was to reduce the number of expired deadlines by observing the environment and updating its density and intensity. From the experimental results, it was possible to test the proposed approach with different DTA methods, and it was found that using multiple collection zones was more suitable for the proposed approach in terms of both the foraging and expiry rates. Although communication was limited, the use of the DD_Model and hotspot intensity gave higher priority to infected resources; the results showed that hotspot DTA performed better when comparing different map setups. In regards to scalability and robustness, it was found that the proposed method produced consistent results, as it tried to maintain a similar performance across changing environments. Thus, unlike the other methods in the literature, it produces flexible, scalable, and robust results.

One important observation that needs to be addressed in the future is that the expiry rate can be relatively high in the experiments. This is because the search is blind and the communication among robots is limited. Thus, a future research direction will focus on the method of selecting the model parameters adaptively, in which the parameters of the DD_Model are automatically updated based on a real environment setup. In addition, ED_Foraging has many potential applications due to its inherent flexibility, scalability, and robustness. Therefore, future work will also involve modeling real-life problems, such as rescuing COVID-19 patients or operating a recycling factory, and testing them with real swarms of robots.

Funding: This research received no external funding.

Acknowledgments: This work was supported by the Deanship of Scientific Research (DSR) with King Saud University, Riyadh, Saudi Arabia, through a research group program under Grant RG-1441-503. The author also thanks the Ibn Khaldun Fellowship program for Saudi Women at MIT.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lu, Q. An Efficient Multiple-Place Foraging Algorithm for Scalable Robot Swarms. Ph.D. Thesis, University of New Mexico, Albuquerque, NM, USA, 2019.
2. Wei, C.; Hindriks, K.V.; Jonker, C.M. Dynamic task allocation for multi-robot search and retrieval tasks. *Appl. Intell.* **2016**, *45*, 383–401. [\[CrossRef\]](#)
3. Lee, W.; Vaughan, N.; Kim, D. Task Allocation into a Foraging Task with a Series of Subtasks in Swarm Robotic System. *IEEE Access* **2020**, *8*, 107549–107561. [\[CrossRef\]](#)
4. Nedjah, N.; Junior, L.S. Review of methodologies and tasks in swarm robotics towards standardization. *Swarm Evol. Comput.* **2019**, *50*, 100565. [\[CrossRef\]](#)
5. Lu, Q.; Fricke, G.M.; Ericksen, J.C.; Moses, M.E. Swarm Foraging Review: Closing the Gap Between Proof and Practice. *Curr. Robot. Rep.* **2020**, *1*, 215–225. [\[CrossRef\]](#)
6. Wei, Y.; Hiraga, M.; Ohkura, K.; Car, Z. Autonomous task allocation by artificial evolution for robotic swarms in complex tasks. *Artif. Life Robot.* **2019**, *24*, 127–134. [\[CrossRef\]](#)
7. Talamali, M.S.; Bose, T.; Haire, M.; Xu, X.; Marshall, J.A.R.; Reina, A. Sophisticated collective foraging with minimalist agents: A swarm robotics test. *Swarm Intell.* **2020**, *14*, 25–56. [\[CrossRef\]](#)
8. Lee, W.; Kim, D. Adaptive approach to regulate task distribution in swarm robotic systems. *Swarm Evol. Comput.* **2019**, *44*, 1108–1118. [\[CrossRef\]](#)
9. Khaluf, Y.; Vanhee, S.; Simoens, P. Local ant system for allocating robot swarms to time-constrained tasks. *J. Comput. Sci.* **2019**, *31*, 33–44. [\[CrossRef\]](#)
10. Harwell, J.; Gini, M. Broadening applicability of swarm-robotic foraging through constraint relaxation. In Proceedings of the 2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), Brisbane, Australia, 16–19 May 2018; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2018; pp. 116–122.
11. Tang, Q.; Yu, F.; Xu, Z.; Eberhard, P. Swarm Robots Search for Multiple Targets. *IEEE Access* **2020**, *8*, 1. [\[CrossRef\]](#)
12. Loli, E.; Piccolomiini Zama, F. Monitoring Italian COVID-19 spread by an adaptive SEIRD model. *medRxiv* **2020**. [\[CrossRef\]](#)
13. Hladish, T.; Melamud, E.; Barrera, L.A.; Galvani, A.; Meyers, L.A. EpiFire: An open source C++ library and application for contact network epidemiology. *BMC Bioinform.* **2012**, *13*, 76. [\[CrossRef\]](#)
14. Pinciroli, C.; Trianni, V.; O’Grady, R.; Pini, G.; Brutschy, A.; Brambilla, M.; Mathews, N.; Ferrante, E.; Di Caro, G.; Ducatelle, F.; et al. ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intell.* **2012**, *6*, 271–295. [\[CrossRef\]](#)
15. Ahmad, A.; Garhwal, S.; Ray, S.K.; Kumar, G.; Malebary, S.J.; Barukab, O.M. The Number of Confirmed Cases of Covid-19 by using Machine Learning: Methods and Challenges. *Arch. Comput. Methods Eng.* **2020**, 1–9. [\[CrossRef\]](#) [\[PubMed\]](#)
16. Manchein, C.; Brugnago, E.L.; Da Silva, R.M.; Mendes, C.F.O.; Beims, M.W. Strong correlations between power-law growth of COVID-19 in four continents and the inefficiency of soft quarantine strategies. *Chaos: Interdiscip. J. Nonlinear Sci.* **2020**, *30*, 041102. [\[CrossRef\]](#) [\[PubMed\]](#)
17. Vanhee, S. Optimising Task Allocation under Deadline Constraints in Swarm Robotics Using the ACO Algorithm. Master’s Thesis, Ghent University, Belgium, Ghent, 2018.
18. Khaluf, Y. Task Allocation in Robot Swarms for Time-Constrained Tasks. Ph.D. Thesis, University of Paderborn, Paderborn, Germany, 2014.
19. Chung, S.-J.; Paranjape, A.A.; Dames, P.; Shen, S.; Kumar, V. A Survey on Aerial Swarm Robotics. *IEEE Trans. Robot.* **2018**, *34*, 837–855. [\[CrossRef\]](#)
20. McLurkin, J.; Yamini, D. Dynamic Task Assignment in Robot Swarms. In Proceedings of the Robotics: Science and Systems I, Cambridge, MA, USA, 8–11 June 2005.
21. Pang, B.; Song, Y.; Zhang, C.; Wang, H.; Yang, R. Autonomous Task Allocation in a Swarm of Foraging Robots: An Approach Based on Response Threshold Sigmoid Model. *Int. J. Control. Autom. Syst.* **2019**, *17*, 1031–1040. [\[CrossRef\]](#)
22. Alshawi, M.; Shalan, M. Minimal time dynamic task allocation for a swarm of robots. *Int. J. Mech. Eng. Robot. Res.* **2017**, *6*, 481–487. [\[CrossRef\]](#)
23. Cai, Y.; Yang, S.X. A PSO-based approach to cooperative foraging tasks of multi-robots in completely unknown environments. In Proceedings of the 2014 World Automation Congress (WAC), Waikoloa, HI, USA, 3–7 August 2014; pp. 813–822.
24. Jevtic, A.; Andina, D.; Jamshidi, M. Distributed Task Allocation in Swarms of Robots. In *Swarm Intelligence for Electric and Electronic Engineering*; IGI Global: Pennsylvania, PA, USA, 2012; pp. 170–193.
25. Lu, Q.; Hecker, J.P.; Moses, M.E. The MPFA: A multiple-place foraging algorithm for biologically-inspired robot swarms. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 3815–3821.

26. Lu, Q.; Hecker, J.P.; Moses, M.E. Multiple-place swarm foraging with dynamic depots. *Auton. Robot.* **2018**, *42*, 909–926. [[CrossRef](#)]
27. Lu, Q.; Moses, M.E. A Bio-Inspired Transportation Network for Scalable Swarm Foraging: Extended Abstract. In Proceedings of the 2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), New Brunswick, NJ, USA, 22–23 August 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 53–55.
28. Song, Y.; Fang, X.; Liu, B.; Li, C.; Li, Y.; Yang, S.X. A novel foraging algorithm for swarm robotics based on virtual pheromones and neural network. *Appl. Soft Comput.* **2020**, *90*, 106156. [[CrossRef](#)]
29. Kurdi, H.; AlDaoud, M.F.; Al-Megren, S.; Aloboud, E.; Aldawood, A.S.; Youcef-Toumi, K. Adaptive task allocation for multi-UAV systems based on bacteria foraging behaviour. *Appl. Soft Comput.* **2019**, *83*, 105643. [[CrossRef](#)]
30. Khaluf, Y.; Birattari, M.; Hamann, H. A swarm robotics approach to task allocation under soft deadlines and negligible switching costs. In *From Animals to Animats 13*; del Pobil, A.P., Chinellato, E., Martinez-Martin, E., Hallam, J., Cervera, E., Morales, A., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 270–279.
31. Khaluf, Y.; Rammig, F. Task Allocation Strategy for Time-Constrained Tasks in Robots Swarms. In Proceedings of the Advances in Artificial Life, ECAL 2013, Sicily, Italy, 2–6 September 2013; The MIT Press: Cambridge, MA, USA, 2013; pp. 737–744.
32. Pinciroli, C.; Trianni, V.; Grady, R.O.; Pini, G.; Brutschy, A.; Brambilla, M.; Mathews, N.; Ferrante, E.; Caro, G.D.; Ducatelle, F.; et al. ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; IEEE: Piscataway, NJ, USA; pp. 5027–5034.