Adaptive task allocation for multi-UAV systems based on bacteria foraging behaviour

As Published	10.1016/J.ASOC.2019.105643
Publisher	Elsevier BV
Version	Author's final manuscript
Citable link	https://hdl.handle.net/1721.1/136384
Terms of Use	Creative Commons Attribution-NonCommercial-NoDerivs License
Detailed Terms	http://creativecommons.org/licenses/by-nc-nd/4.0/

Adaptive Task Allocation for Multi-UAV Systems based on Bacteria Foraging Behaviour

Heba Kurdi[a,b,], Munierah F. AlDaood[a], Shiroq Al-Megren[b,c], Ebtesam Aloboud[d], Abdulrahman S. Aldawood[e], Kamal Youcef-Toumi[b]

[a] *Computer Science Department, King Saud University, Riyadh 11495, Saudi Arabia*
[b] *Mechanical Engineering Department, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
[c] *Information Technology Department, King Saud University, Riyadh 11495, Saudi Arabia*
[d] *Computer Science Department, Al Imam Mohammad Ibn Saud University, Riyadh 11432, Saudi Arabia*
[e] *Department of Plant Protection, King Saud University, Riyadh 11495, Saudi Arabia*

Abstract

The foraging behaviour of bacteria in colonies exhibits motility patterns that are simple and reasoned by stimuli. Notwithstanding its simplicity, bacteria behaviour demonstrates a level of intelligence that can feasibly inspire the creation of solutions to address numerous optimisation problems. One such challenge is the optimal allocation of tasks across multiple unmanned aerial vehicles (multi-UAVs) to perform cooperative tasks for future autonomous systems. In light of this, this paper proposes a bacteria-inspired heuristic for the efficient distribution of tasks amongst deployed UAVs. The usage of multi-UAVs is a promising concept to combat the spread of the red palm weevil (RPW) in palm plantations. For that purpose, the proposed bacteria-inspired heuristic was utilised to resolve the multi-UAV task allocation problem when combatting RPW infestation. The performance of the proposed algorithm was benchmarked in simulated detect-and-treat missions against three long-standing multi-UAV task allocation strategies, namely opportunistic task allocation, auction-based scheme, and the max-sum algorithm, and a recently introduced locust-inspired algorithm for the allocation of multi-UAVs. The experimental results demonstrated the superior performance of the proposed algorithm, as it substantially improved the net throughput and maintained a steady runtime performance under different scales of fleet sizes and number of infestations, thereby expressing the high flexibility, scalability, and sustainability of the proposed bacteria-inspired approach.

Keywords: Task allocation, unmanned aerial vehicles, distributed systems, multi-UAV systems, bacteria inspiration, optimisation problem.

1. Introduction

The devastating economical impact of red palm weevil (RPW, Rhynchophorus ferrugineus - Curculionidae: Coleoptera) infestation is recognised globally, as it affects various palm species in 54 countries [1]. Mature beetles burrow to a palm's stem and crown where they lay their eggs. At its larvae stage, an RPW is at its most damaging stage, as it prospers within the palm, damaging its vascular system [2, 3]. Early detection is difficult due the cryptic feeding behaviour of the larvae and the palm's lack of visual symptom of infestation. Nevertheless, early detection is crucial to combating the spread of the infestation when it can still be treated by pesticides. Several techniques are used to ease detecting RPW infestations, including chemo/olfactory sensing by dogs, X-ray imaging, thermal imaging, and acoustic detection of RPW larvae [4, 5, 6]).

The use of fleets of unmanned aerial vehicles (UAVs) is becoming increasingly compelling for detect-and-treat (DAT) missions, where they are utilised to assist in eradicating agricultural pests (e.g., [7, 8, 9]) and are continuously being improved, for instance, in terms of reducing communication disturbances and improving synchronisation (e.g., [10, 11]). One of the major challenges to this approach is the optimal partitioning of tasks across UAVs such that the objective of the DAT mission is optimised. As a special case of the NP-hard multi-robot task allocation (MRTA) problem, the multi-UAV task allocation (MUTA) problem has attracted notable attention and is often approached using heuristics because of their fast development and ease of application [12]. However, unlike optimal algorithms, heuristics are best assessed empirically under controlled experimental conditions [13].

NP-hard problems, such as MRTA and MUTA, are typically approached using heuristics that suggest 'good' solutions such as max-sum, auction-based, and bio-inspired algorithms (e.g., [14, 15, 16]). The majority of such approaches can be categorised as problem-independent heuristics, i.e., metaheuristics. A metaheuristic starts with a random solution that is iteratively optimised until a near-optimal solution is reached. Implementing a metaheuristic is relatively easy because one algorithm can be utilised for every agent, where parametrisation and neighbouring strategies can also be used to reflect different behaviours. However, the iterative nature of metaheuristics increases the power and time consumptions, thereby significantly impacting the overall performance of the algorithm. The solution proposed in this paper falls under the second category, i.e., it is a problem-dependent heuristic. While more difficult to implement (multiple algorithms), a problem-dependent heuristic is tailored to the specific problem, and its best-effort approach attempts to achieve a good guess sans iterative improvements [17]. The choice between problem-dependent and problem-independent heuristics is dependent on the nature of the optimisation problem and the trade-off between the cost of implementation and runtime operation.

Optimisation problems can grow more difficult in a distributed setting and are known as distributed constraint optimisation problems (DCOPs). The max-sum metaheuristic is an approach that provides satisfactory approximate solutions to challenging decentralised optimisation problems [18]. The metaheuristic approaches the optimisation problem by breaking its maximising function iteratively into a sum of smaller functions, where agents maximise the global utility based on their variables and constraints [19]. Max-sum has been applied to many UAV application domains (e.g., [20, 14]); however, the algorithm's exponential running time, $O(m^n)$ (where m is the number of agents and n is the number of tasks), and lack of support for situational awareness are its main drawbacks. This is because the max-sum algorithm does not take the dynamic environment into account nor does it consider multiple task allocation objectives; thus, solutions are re-calculated in their entirety to account for changes in the environment [21].

Auction-based heuristics offer a less expensive alternative to DCOPs and decentralised decision making [22, 23, 16]. Auction-based heuristics approach the MUTA problem by offering tasks for auction, where UAVs bid for allocation pertaining to the cost of running the task. To solve conflicts and determine the winning bid, a consensus approach is often implemented. The adoption of auction-based allocation results in numerous issues such as the complexity introduced by auctioning off new tasks. The heuristic is also reputed to be time and resource consuming, as bids are calculated by each UAV and a winning bid is chosen; resulting in quadratic time complexity $O(n^2m)$, where m is the number of agents and n is the number of tasks [24].

Bio-inspired algorithms are based on the natural behaviour of simple agents as they interact amongst themselves and exhibit favourable features, such as self organisation, adaptiveness, and robustness [25]. This natural behaviour is governed by simple rules that support their implementation and economical execution. The majority of bio-inspired algorithms are concrete mathematical and probabilistic models that quantify natural features that are based on a certain set of predefined assumptions (e.g. [26]). Nevertheless, there has been a growing interest in utilising problem dependent heuristics for overcoming complex optimisation problems. Recently, LIAM was presented as a locust-inspired problem-dependent heuristic to optimise the allocation of multi-UAVs in SAR missions [27, 17]. Locusts in nature dynamically adapt to internal and external stimuli between solitary and gregarious roles. LIAM mimics the adaptive behaviours of locusts, which are demonstrated in the system's fully autonomous UAVs. The performance of LIAM was comparatively assessed against auction, max-sum, ant colony optimisation (ACO), and opportunistic task allocation (OTA), where LIAM was proven to be superior given the dynamic nature of SAR missions with a higher net throughput and a shorter mean rescue time.

The simple behaviour of bacteria has continually been an inspiration for computational practices and optimisation heuristics. Specifically, two heuristics, bacteria foraging optimisation (BFO) [28] and bacterial chemotaxis (BC) [29], have broadened the application scope of bacteria optimisation. The BFO algorithm is inspired by the social foraging behaviour of bacteria and was applied to the continuous function optimisation problem domains [30]. In the same year, the bacterial chemotaxis (BC) heuristic simulated the movement of a single bacterium and evolutionary concepts to improve optimisation strategies and problem-solving capabilities. Since the development of BFO and BC, there has been a growing number of extensions that attempt to hybridise the algorithms with other metaheuristics and computational intelligence algorithms (e.g., [31, 32, 33, 34, 15]).

The bacterial colony chemotaxis (BCC) algorithm is based on the information interactive model between bacteria via chemo-attractants [35]. Several assumptions are made about bacterial colony correspondence: locomotion self-regulation based on information from approximate bacteria and migration simulations. The algorithm has been applied to solve optimisation problems in various fields, e.g., machine learning and inverse air-foil design [36]. A scheduling algorithm, the super-bug algorithm (SuA), was similarly inspired by bacteria [37], in particular, the antibiotic resistance developed from bacterial mutation. SuA was comparatively assessed against other techniques on the flexible manufacturing scheduling problem, where it performed better in most cases. The viral infection process motivated the proposal of the viral system algorithm, which consists of replication and infection mechanisms [38]. These mechanisms are used to generate meta-heuristics that overcome computer security problems and virus elimination.

Inspired by bacteria-based algorithms, this paper presents a multi-UAV task allocation for RPW combat based on bacteria behaviour (UTARB) algorithm. Its main advantages over other MUTA approaches is the adaptive behaviour of the UAVs and the autonomic nature of control and decision making. Unlike other approaches, UTARB targets non-clairvoyant tasks and is tailored to the particularities of DAT missions and tasks. By adopting a problem-dependent heuristic approach, the proposed model ensures that the optimization algorithm is easier and quicker to implement and more robust in its adaptation. This is made possible by the algorithm as computationally expensive parameters are ignored, while simpler parameters that are indirectly correlated with system performance are relied upon [39, 13]. Accordingly, the proposed approach does not aim to prove a first-order relationship between the proposed heuristic and the desired results.

A simulation model was built for DAT missions to thoroughly assess the performance of the proposed algorithm and three long-standing heuristics: auction-based, max-sum, and opportunistic coordination schemes, as a well as a recently introduced locust-inspired heuristic for multi-UAV task allocation in search and rescue missions (LIAM) [17]. The experimental results demonstrated the superiority of UTARB over the benchmark algorithms, where it yielded a significant increase in the percentage of detected infestation at reduced runtimes. This paper extends on previous work [40] that highlight the development of UTRAB to further signify its efficiency.

The main contributions of this paper can be summarised as follows:

- A new bacteria-inspired heuristic for MUTA problems, UTARB, is proposed. The proposed algorithm is a problem-specific heuristic inspired by the foraging behaviour of bacterial colonies for addressing the special challenges of DAT missions.
- A well-controlled experimental framework for evaluating UTARB in DAT missions is developed.
- A thorough investigation of the performance of UTARB is conducted against well-established benchmark algorithms and a problem dependent bio-inspired algorithm with different numbers of infested palms and deployed UAVs.

The remainder of this paper is organised as follows. Section 2 reviews existing work regarding the scheduling problem inspired by the behaviour of fish and bio-inspired heuristics that were developed to address the problem of multi-robot or multi-UAV task allocation. The following section, Section 3, describes the bacteria inspiration behind this work. Section 4 illustrates the system model. Section 5 introduces UTARB's search algorithms. Section 6 describes the experimental configuration and procedure for conducting the simulations. Section 7 presents and explains the results of the comparative evaluations. Finally, Section 8 concludes the paper and briefly discusses future work.

2. Related Work

The past few decades have seen a shift of focus in the field of robotics toward investigating problems of coordinating multiple robots. This is due to the increasing complexity of multi-robot and multi-UAV systems as fleets expand in size while agents and tasks increase in heterogeneity. As a result, the problems of multi-robot and multi-UAV coordination have received significant attention. This section reviews numerous works that address the MRTA and MUTA problems in various scenarios. The review is not meant to be exhaustive, as it focuses on solutions proposed by the benchmark algorithms (such as auction and max-sum heuristics) and biologically inspired heuristics.

The max-sum algorithm was initially proposed for DCOPs in multi-agent systems [41] and has since been utilised to address several other optimisation problems such as the allocation of tasks to multi-UAV fleets. Agents in max-sum-based approaches can generate a consistent task allocation plan by exchanging and adjusting the utility function, thus enabling cooperation when performing tasks. A max-sum coordination mechanism was proposed for a fleet of autonomous UAVs as they survey a disaster area to provide aerial images [42, 43]. The task allocation problem is addressed asynchronously, by which a computed utility value is maintained by the UAVs for each task. The proposed model was assessed and exhibited promising potential, as it provided a favourable trade-off between the quality and quantity of tasks performed. Hardware tests were also conducted to assess the coordination mechanism proposed using commercial off-the-shelf hexacopter UAVs deployed in the real world, the results of which confirmed the performance of the max-sum algorithm in coordinating UAVs in real-word situations. Nevertheless, the empirical tests were not time constrained and were restricted to ten surveying UAVs in a limited disaster area. The traceability of this solution increases in difficulty as the number of tasks and agents sufficiently increases due to the exponential number of constraints.

For urban disaster environments, a binary max-sum algorithm was proposed for clustering-based task allocation [21]. Tasks in the proposed algorithm are distributed using a distance metric between the agents' features and tasks, along with the benefits of facto graphs with THOPs to optimise a global objective function. The modelled heuristic was comparatively assessed against an optimised multi-team task allocation model. The result of the assessment demonstrated the algorithm's superiority, as it reduced communication costs and non-concurrent constraint checks. The max-sum algorithm was also adapted for decentralised coordination for the purpose of considering constraints imposed by a human operator [44]. The algorithm supports accountability for both human- and agent-based decision making by providing a fully tracked provenance infrastructure in a disaster management system prototype. Assessments were performed to address the interaction between agent and human operators, where the system demonstrated improved performance as strong control is given to the user over autonomy when allocating tasks.

Auction-based algorithms are one class of decentralised combinatorial algorithms that have been utilised to efficiently produce suboptimal solutions for the allocation of tasks over a team of agents. The algorithm is at times augmented with a consensus protocol to resolve assignment conflicts among agents. One such example is a consensus-based auction algorithm (CBBA) that was proposed to negotiate between agents by forming an initial greedy task allocation [45]. CBBA utilises an auction approach and a consensus procedure for task selection and conflict resolution, respectively. The final task allocation is determined by achieving a consensus on the winning bid, thus reducing computation costs and improving convergence. Experiment were performed to assess the performance of CBBA against an existing sequential auction algorithm. The performance of CBBA proved to be superior, showing better convergence properties. Clustering-based task allocation was proposed as a simulation model involving task priority and balancing in multi-robot systems [46]. The model attempts to obtain a balanced exploration path by considering the costs of robot travel and idleness, thus minimising the average waiting and completion times. The proposed methodology consists of K-means clustering and auction-based mechanisms to achieve an appropriate balance. Increased cluster numbers were found to significantly affect total cost, and further studies are required to analyse their effect.

Several auction-based approaches were compared to assess their performance for the efficient allocation of tasks for multi-robot teams in a dynamic environment [22]. The effectiveness of the auction mechanisms in this study considered the total distance of travel, cost of task execution, as well as how well the task was executed. More recently, a cooperative rescue plan for search and rescue missions is devised using an auction-based allocation approach [16]. The proposed auction approach is used to best allocate tasks to rescue teams for the purpose of enhancing cooperation. A landslide post disaster environment was built to demonstrate the performance of the proposed algorithm with a non-cooperation rescue plan and F-max-sum. Findings show that the proposed auction-based approaches increases the ratio of rescued survivors and the probability of survival. Additional evaluations were conducted to determine the robustness and sensitivity of the proposed solution. Robustness analyses have shown that efficiency is significantly affected by the search radius and, thus, that a high level of cooperation should be maintained. The sensitivity analysis identified trade-offs between cooperation, independent rescue searches and search coverage that greatly affect rescue efficiency.

The collective behaviours of social insect colonies and animal groups are characterised by self-organised control and collaboration, elastic properties and effective interaction schemes [47]. These behaviours analogically align themselves with distributed optimisation problems such as task allocation and path planning. Bio-inspired algorithms that address the task allocation problem considerably diverge depending on the species that they simulate. These include, but are not limited to, algorithms inspired by the foraging behaviours of locusts [17], ants [26] and honeybees [48] and the brood parasitism of cuckoos [49].

Locusts exhibit adaptable morphological and behavioural forms as they advance through their lifecycle. Their behaviour is dramatically altered in response to internal and external stimuli between two main phases: solitarious and gregarious. Although there are few heuristics inspired by this type of behaviour, a locust-inspired algorithm for task allocation in a multi-UAV distributed system performing SAR missions was recently proposed [27]. Locust behaviour ideally maps to the main roles of SAR missions, search and rescue, which behaviourally imitate solitarious and gregarious locusts, respectively. The effectiveness of the locust-based algorithm was compared against OTA in terms of net throughput, survivor rescue time, and runtime performance. The findings revealed the proposed algorithm's superior net throughput compared to the benchmark. The work was recently extended to best demonstrate the adaptive behaviour of autonomous UAVs in multi-UAV SAR missions [17]. The locust-inspired task allocation (LIAM) algorithm was extensively tested under various conditions of area scale, numbers of survivors, and the size of the multi-UAV fleet. The experimental results demonstrated the superiority of LIAM compared to well-established algorithms, primarily auction, max-sum, ACO, and OTA algorithms. The proposed algorithm maintained a significantly higher percentage of rescued survivors and reduced task completion time.

ACO is based on the behaviours of ant colonies as they forage for food. Ants leave a trail of chemical pheromones to guide other ants to discovered food sources; paths with strong concentrations of pheromones are given priority to support the search for the shortest path between the colony's nest and the food source [26]. ACO was proposed for MRTA, which utilises two ant colony processes that use pheromones to allocate tasks to robots and determine each robot's task processing sequence [50]. A simulation environment was built for multi-robots transporting containers at a dock, where the proposed ant-inspired algorithm was comparatively tested against another algorithm that unified the problems of task allocation and path planning. Several comparative scenarios were implemented with a variable number of containers. The processing time of the proposed algorithm was comparatively short, likely due to its simultaneous scheduling capability.

The ACO algorithm was also adapted to combat the task allocation problem in multi-agent environments, thereby reducing processing times and achieving global optimisation [51]. The proposed algorithm, collection path ACO (CPACO), extends ACO by modifying the heuristic by establishing a 3-dimensional pheromone path to resolve the MUTA problem. The performance of CPACO was comparatively assessed against gravitational search and the forward optimisation heuristic. Although CPACO consumed more time than the forward optimisation heuristic, CPACO's efficiency was substantially better. Nonetheless, the proposed algorithm was sensitive to the initial parameters and the number of ants to convergence. The dynamic ant colony labour division (DACLC) algorithm was proposed to solve the task allocation problem in a dynamic battlefield with a swarm of combat multiple UAVs [52]. The proposed algorithm is based on the fixed response threshold model (FRTM) that addressed the problem of adapting the system to various demand levels. The effectiveness of DACLD was determined in several simulated scenarios with varying numbers of targets and emerging threats. DACLD was found to be robust and flexible in its dynamic allocation of tasks with a high degree of self-organisation. DACLD remains to be compared against well-established algorithms.

As prokaryote, bacteria behave simply and in patterns that can easily be described and mimicked computationally. The typical behaviours of bacteria during their lifecycle, i.e., chemotaxis, communication, reproduction, and migration, have inspired several general optimisation algorithms (e.g., BFO [28] and BC [29]) that have since been applied to multi-robot mission optimisation. An adapted implementation of BFO was proposed for a multi-robot search and mapping of chemical gas concentration [53]. The robots in the proposed algorithm perform the search autonomously via bacterial chemotaxis behaviour and send their sensed data to a ground station. In contrast to the artificial bacteria behaviour in BFO, robots have continuous dynamics and traverse all the paths between its current position towards a different location. Therefore, the adapted BFO algorithm generates high-level path planning and waypoints, which are used iteratively to compare chemical concentrations. The data received are then combined, interpolated, and filtered to form a real-time map of chemical gas concentrations in an environment. The performance of the implemented BFO was later evaluated against other bio-inspired implementations (ACO and decentralised asynchronous particle swarm optimisation), sweeping, and canonical particle swarm optimisation [54]. The findings demonstrate the superior performance of the bio-inspired implementation for concentration map building, where BFO and ACO were able to complete their search.

A multi-robot path planning algorithm inspired by the original BFO for known and unknown targets was developed [55]. A clustering-based method was used to divide the area virtually, and bacteria-inspired direction-based movement was utilised. The algorithm was tested for simple and complex environments, where the results showed that the proposed algorithm was able to efficiently perform path planning to the classified targets. Similarly, the BFO algorithm was applied to the problem of mobile robot navigation to determine the shortest path to a target position in an unknown environment with moving obstacles [56]. Particles are randomly distributed around a robot, by which the best particle is selected based on the distance to the target and a cost function. The selection of the best particle is generated using a high-level decision strategy, and the robot proceeds to its target.

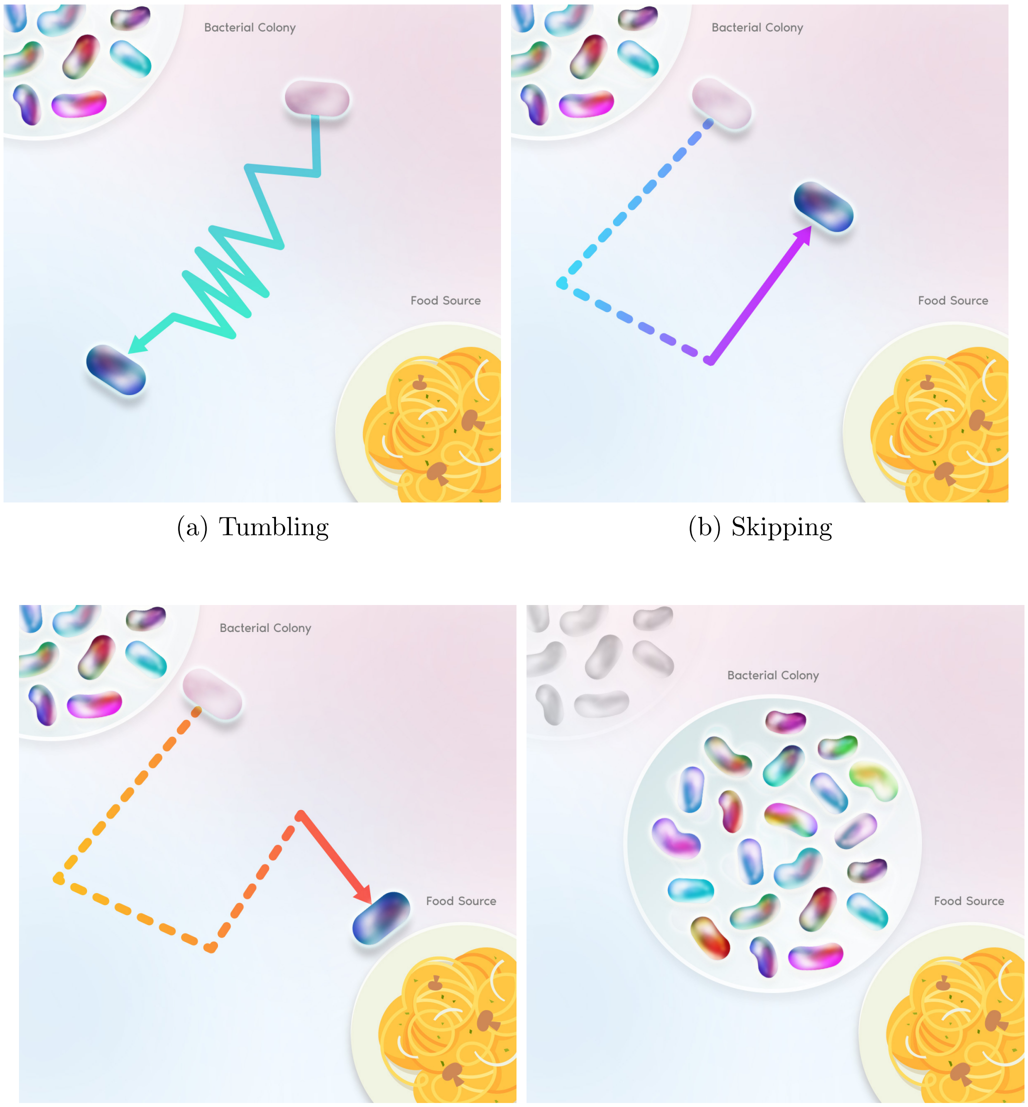
The efficiency of the proposed approach was assessed against the standard BFO and particle swarm optimisation. The findings showed that the proposed bacteria- and particle-inspired algorithm produces better solutions and optimal paths.

A self-organisation algorithm inspired by the behaviour of bacteria was proposed for multi-robot target search and trapping [57]. Target search and trapping tasks were achieved by the robots using bacteria chemotaxis guided by the gradient information obtained from the target until the target was located. Simulations were conducted to assess the performance, robustness, and complexity of the proposed algorithm. The findings demonstrated the effectiveness of the algorithm and robustness under unanticipated failures. The results also proved the algorithms ability to avoid being trapped in local optima and its computational efficiency. The BFO algorithm was also the source of inspiration and improvement for multi-robot cooperation for nanarobotics and nanomedicine [58]. Referred to as the improved BFO algorithm (IBFOA), the proposed methods utilised cooperative learning for multiple nanorobots to reach and eradicate cancer cells in a blood vessel. The performance of the proposed IBFOA algorithm was compared against the standard BFO algorithm. The findings demonstrated the superior performance of IBFOA, as it reduced time complexity and improved global search.

Previous work, be it based on conventional approaches (e.g., auction and max-sum) or swarm intelligence, would often introduce general optimisation solutions for the MUTA problem (i.e., problem-independent heuristics and metaheuristics) rather than tailored algorithms that consider the particularities of the mission or task (i.e., problem-dependent heuristics). Moreover, to the best of our knowledge, the behaviour of bacteria has not been previously adapted for non-clairvoyant MUTA tasks where job information is unknown a priori.

3. Bacteria Inspiration

Natural systems consist of simple agents that interact with each other by abiding to simple rules. These agents are self-organised and are capable of adapting to changing requirements and environments. These properties enable natural systems to solve complex problems that are beyond the abilities of current computer systems [59, 60, 61]. The simple agents and rules of natural systems of many organisms have been simulated to govern task allocation (e.g., [17, 52]), resource management (e.g., [62, 63]), synchronisation (e.g.,



(a) Tumbling (b) Skipping

(c) Swimming (d) Swarming

Figure 1: Bacteria foraging behaviour as a bacterium moves from a starting to an end position with lines depicting its movement.

[64, 65]), and social differentiation (e.g., [66, 67]). For many swarm organisms, the foraging process involves the aggregation of organisms into groups to search for sustenance by maximising the energy obtained per unit time

spent foraging [68]. The aggregation of organisms into social foraging groups is also a key element for avoiding predators and increasing their chances of finding profitable food sources [69]. Bacterial colonies exhibiting movement during foraging show some intelligence that cannot be simply regarded as random or arbitrary [60, 70, 30, 68].

The mobility of bacteria can be divided into four types, as shown in Figure 1: tumbling, skipping, swimming, and swarming. In the absence of stimuli, a bacteria cell forages for nutrients in random directions; it moves in a straight line for some time, then changes its angle and repeatedly moves in a certain pattern for an arbitrary amount of time and as along as no stimulus is experienced. This tumbling behaviour serves to randomly reorient the bacteria. Bacteria typically alternate periods of tumbling and swimming. In the latter instance, the bacteria is in the presence of a stimulus and thus moves directly towards it. Tumbling and swimming are collectively known as chemotaxis, where bacteria direct their movement based on the presence of chemical gradients, i.e., stimuli [69, 71]. As a bacteria cell tumbles for a long period of time without encountering a stimulus, it significantly changes its direction by skipping towards a new course. Bacteria secrete attracting and repelling chemicals into the environment to communicate. If a food source is found, a bacterial cell releases attractants for other bacteria to sense and swarm around the source. If a stimulus is found to be revolting, the bacteria cell releases repellents for others to keep away from the stimulus [72, 60]. This continuously changing strategy for selecting a search point based on the current situation is the source of inspiration for this work.

4. System Model

When building a system model, the system is simplified to its main properties and functions. Nominally, system models are classified as: (1) mathematical or analytical models, and (2) simulation models. Mathematical models provide an abstraction of the system, which are represented as equations that summarise the system performance. Simulation models, on the other hand, experimentally mimic events that occur in the real system. Thus, allowing experimentation with different parameters and control logic [73].

Several works that mathematically model the foraging behaviour of bacteria have already been published and reviewed in Section 2. These models quantify the features of the foraging behaviour of bacteria colonies based on a set of predefined assumptions. The rigidity of these mathematical approaches are built upon assumptions that are different from the unique characteristics of biological systems. For instance, analytical models measure the system behaviour using expected values for a predefined set of performance metrics, which ignore any changes in the system behaviour over time [74]. This disregards the flexibility of natural systems that typically employ an adaptive control strategy to persevere in a changing world.

This paper presents a generic model for UTARB that is simulated in several representative scenarios. The proposed system is comprised of a finite number of UAVs in a multi-UAV fleet and a finite number of infested palms in a plantation area. Each UAV is capable of executing one task at a time. The goal of UTARB is to assign detect-and-treat tasks to UAVs to maximise the overall number of detected and treated infested palms, considering the urgency of the infested case and efficiency of the UAVs in conducting the task.

The proposed system is comprised of three main components:

- The *palm plantation* is a physical agricultural field that consists of palm trees that are to be investigated for RPW infestations.
- In almost all multi-UAV missions, a *ground station* is required to control and manage the running of the mission. In the case of UTARB, the role of the ground station is limited to sending the mission requirements and the variables necessary for initialising the mission. Upon mission completion, a mission report detailing the outcome is sent to the ground station.
- The *UAVs* are a group of agricultural systems with built-in capabilities for flying and collision avoidance. The agents are equipped with various sensors such as piezo electric microphones, pesticide containers and injectors. UAVs are programmed offline and autonomously fly to detect infestation and treat palms as needed.

The simulation environment considers a DAT mission with the number, severity level, and locations of infested palms unknown a priori. Each infested palm is randomly assigned a severity level, which deteriorates as the mission time passes. Infested palms are scattered across the plantation area arbitrarily, with some hotspots representing areas with a potentially greater concentration of infested palms. An infested palm can be in one of two states:

- A palm with a *mild* infestation that can be remedied using the UAV's built-in pesticide injectors. A mild infestation deteriorates to severe if not detected within a certain time period.

- An infested palm with a *severe* infestation which requires contact with the ground station to call for help.

The plantation area is divided into logical blocks. Each set of blocks in a row is referred to as a *region*. Each block has a size of $a \times b$, where a and b are positive integers. The size of the block is determined during initialisation and assumes one palm per block. Each block can be in one of the following states:

- An *unchecked* block has yet to be checked for infestation by a UAV. This is the initial state for all blocks.

- A *healthy* block contains a palm that is free of infestation.

- A *treated* block is comprised of a mildly infested palms that has since been treated by a UAV.

- A *severe* blocks has a severely infested palm that has been detected by a UAV and the ground station has been contacted for help.

- A *skipped* block refers to a block that has been temporarily bypassed since a UAV assigned to the region has scanned a certain percentage of the region ($P\%$) and no infestations were detected in the other blocks.

Blocks that are labelled as *unchecked* or *skipped* can further be categorised based on their level of urgency. The urgency of a block is set by the UAV after discovering an infested palm in a neighbouring block. The level of urgency for *unchecked* and *skipped* blocks are set as follows:

- An *unchecked* or *skipped* block urgency is set to *very urgent* when a severely infested palm is discovered in its vicinity. In other words, if a palm is discovered to be severely infested, the the urgency level of its direct neighbours (i.e., the eight blocks surrounding the infested block) are set to *very urgent*.

- An *unchecked* or *skipped* block urgency is set to *urgent* when a mildly infested palm is discovered in its vicinity.

Urgency levels are useful, as they allow the UAVs to target the infested region mimicking the behaviour of bacterial swarming.

A region is comprised of a row of blocks and the status of a region is dependent on the status of its blocks. The state changes as the UAVs explore blocks and regions and act accordingly. A region can be in one of the following states:

- A region's state is *unchecked* when all of it blocks are similarly *unchecked*. This is the initial state for all blocks and regions.
- A region is *pending* when at least one block in the region is *skipped* or *unchecked*.
- A region's state is *checked* once all of its blocks have been checked for infestation.

Figure 2 illustrates UTARB's abstract system architecture. The figure shows a plantation area with a number of healthy and infested palms. Region A is marked as *pending* as it contains one *skipped* block that was skipped by the UAV since a consecutive percentage of blocks were labelled as *healthy*, i.e. the palms were not infested. In Region B, a UAV treated three mildly infested palms (B1, B3, B4) and one block was labelled as *severe* and awaiting elimination, thus, the region is marked as *checked*. Since block B2 was labelled as *severe*, all *skipped* and *unchecked* neighbouring blocks were tagged as *very urgent*, such as blocks C1 and C2. This indicates that priority should be given to blocks marked as *very urgent*. An *unchecked* block, C3, in Region C contains a mildly infested palm that is about to be treated by the UAV. Once the palm is injected with the pesticide, block C3 will be labelled as *treated*. Subsequently, all *skipped* and *unchecked* neighbouring blocks to C3 were tagged as *urgent* (i.e. D3 and D4 in Region D).

The wide objective of UTARB is to maximise the net throughput with minimal cost to the running time of the algorithm, which ensures the probability of halting the spread of RPW infestation. For simplicity, it is assumed that several charging and pesticide filling stations are available throughout the plantation to ensure infinite battery lifetime and immediate pesticide refills. These are valid assumptions, as RPW DAT missions are not time critical and resources are usually easy to deploy and reach compared to search-and-rescue and military missions.

Figure 2: The UTARB system abstract architecture.

5. Proposed Method

The task allocation algorithm is the core component of the UTARB system. A task allocator component runs in each UAV to allow for autonomous distributed decision making when allocating tasks, i.e., which block should be assigned to which UAV and when. To achieve this, the mission time is divided into two phases: exploratory search and extensive search. The DAT mission begins by receiving specifications from the ground station so that each UAV can be initialised according to the mission requirements.

5.1. Exploratory Search Algorithm

Exploratory search is the first phase in the DAT mission. This phase implements the four main procedures that mimic the behaviour of mobile bacterial cells as they forage for food (see Section 3). These behaviours can be summarised in the exploratory search phase as follows:

- Tumbling (random selection): The tumbling behaviour of bacteria is mimicked by UAVs at the start of the mission. In this case, UAVs select unchecked regions randomly.

- Skipping: The skipping bacteria behaviour is mimicked in the algorithm by skipping *unchecked* blocks when a certain consecutive percentage ($P\%$) of the blocks was healthy.

- Swimming: After a region is selected by a UAV, the UAV scans the region's block consecutively from left to right or right to left (depending on the block's location). This procedures is similar to that of the swimming behaviour of bacteria as they respond to a stimulus.

- Swarming: In the exploratory search algorithm, swarming is performed implicitly by UAVs without team communication. This is possible with the urgency labels attached to blocks. In this case, blocks in close vicinity to infested blocks attract UAVs due to the higher probability of infestation. This procedure closely mimics the swarming behaviour of foraging bacteria.

These procedures are illustrated in the exploratory search flowchart in Figure 3 and presented as pseudocode in Algorithm 1.

As depicted in Figure 3 and Algorithm 1, the exploratory phases of the DAT mission commences with each UAV randomly selecting a block to explore. The selection is typically based on the urgency level of a block, which is not the case when the mission first starts and the urgency is not yet determined. The UAVs check the infestation level of each block and respond accordingly. If a palm is infestation free, then the block is tagged as *healthy*. A mildly infested palm is treated by the UAV and tagged as such. Additionally, the urgency level of the neighbouring blocks of the mildly infested palm is set to *urgent*. If a palm seems to be severely infested, then the UAV calls the ground station for help, and the block's state is set to *severe*. The urgency level of all neighbouring blocks is also updated to *very urgent*.

The skipping behaviour of bacteria is mirrored early on to determine if a certain consecutive percentage ($P\%$) of the region has been scanned without infestation. This is because the likelihood of the remainder of the region being infestation free is relatively high, and thus, the remainder of the blocks in the region are skipped. If this is not the case, the bacterial swimming behaviour is adopted to move to the next *unchecked* block in the region. Once all blocks are checked in the region, the UAV will scan an *unchecked* region if available

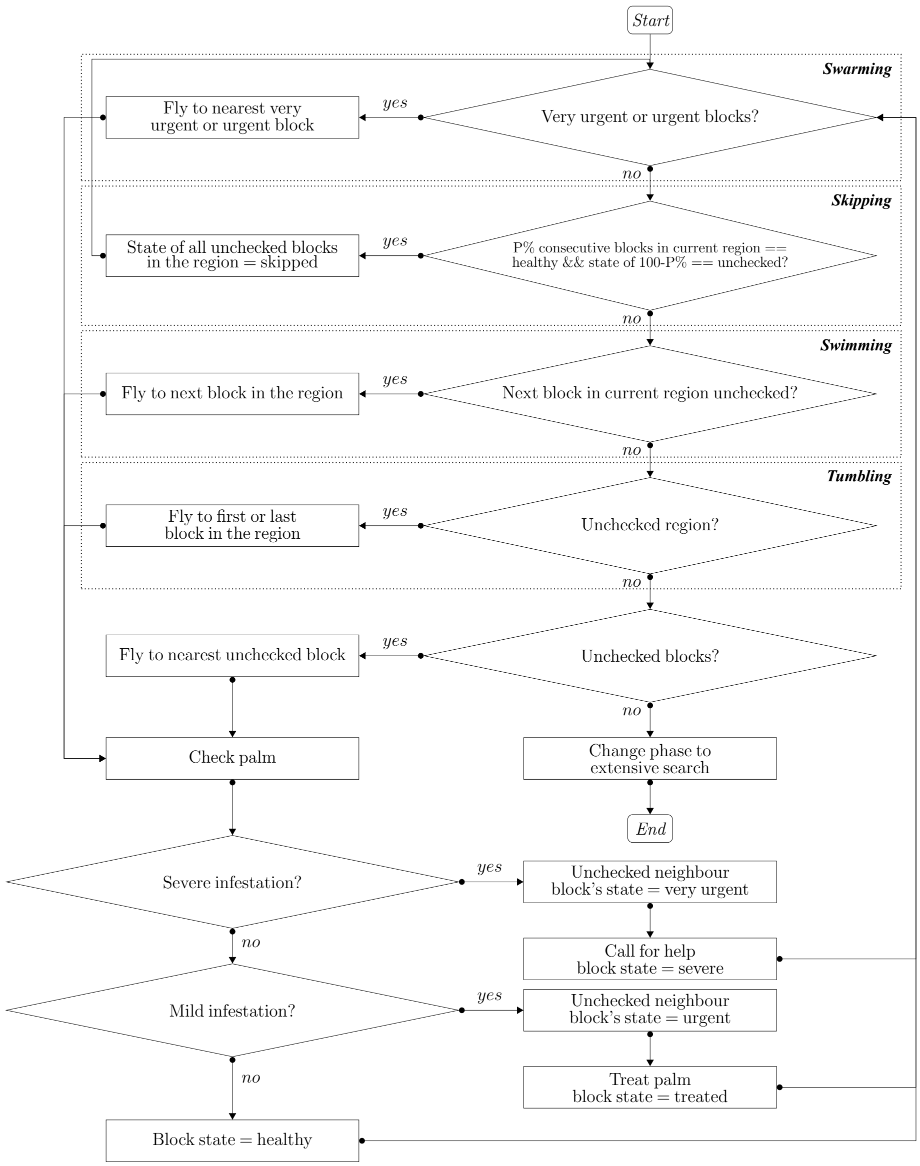


Figure 3: The UTARB system exploratory search flowchart.

Algorithm 1: The proposed algorithm's exploratory search.

```
repeat
    if there are any very urgent or urgent blocks then
        Fly to nearest very urgent or urgent block                //Swarming
    else if state of P% consecutive block in current region is healthy and state of 100 − P% of
      block is unchecked then
        State of all unchecked blocks in the region = skipped     //Skipping
    else if there is an unchecked block in current region then
        Fly to the next block in the region                       //Swimming
    else if there is an unchecked region then
        Fly to first or last block in the region                  //Tumbling
    else if there are unchecked blocks then
        Fly to nearest unchecked block
    else if there are no unchecked blocks then
        Change phase to extensive search
    After selecting a block to detect, check palm
    if palm is severely infested then
        Unchecked neighbors' blocks state = very urgent
        Call for help
        Block state = severe
    else if the palm is mildly infested then
        Unchecked neighbors' blocks state = urgent
        Treat palm
        Block state = treated
    else
        Block state = healthy
until no unchecked blocks remain
```

(i.e., tumbling behaviour), or it randomly selects an *unchecked* block in other regions. This process continues until all blocks have been checked.

Once a palm is reached via the swarming, skipping, swimming or tumbling behaviour, it is then checked for infestation. This, in turn, determines the state of the palms and urgency levels for neighbouring palms. The exploratory phase concludes with at least one skipped block. At this time, the UAVs switch to the extensive search phase.

5.2. Extensive Search Algorithm

In the extensive search cycle, skipped blocks are checked for infestation. The task allocation in the extensive search phase is significantly simpler (see Figure 4 and Algorithm 2). A greedy approach is utilised, where each UAV selects the nearest skipped block to check and tag according to the palm's state.

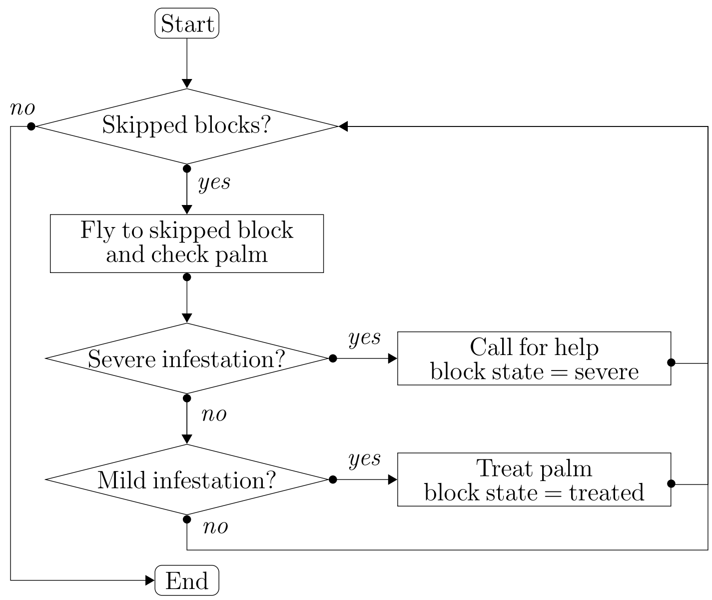


Figure 4: The UTARB system extensive search flowchart.

6. Evaluation

The paper aims to introduce UTARB as a bio-inspired algorithms for DAT missions. The primary hypothesis is that relative to the benchmark algorithms, utilising UTARB in DAT missions will maximise the net throughput without cost to the running time of the algorithm at various problem scales in terms of the number of UAVs and infestations with a set mission time. A well controlled empirical evaluation framework was developed to evaluate the performance of the proposed algorithm and test the hypothesis. A flexible Java simulator (MASPlanes++) was built based on MASPlanes multi-UAV simulator [75] using Java SDK 1.6 and Maven 2.0. MASPlanes++ allows for dynamic illustrations of multi-agent missions that are composed of two groups of tasks, such as is the case for DAT missions. The simulations were conducted on a computer with an Intel Core i7 processor at 2.8 GHZ and paired with 16 GB of RAM operating at 1866 MHz.

The evaluation framework considers the scalability of the proposed and benchmark algorithms to a larger number of UAVs, and the algorithms sustainability under larger number of infestations. For this purpose, two context parameters are controlled in the simulations to represent samples of DAT

Algorithm 2: The proposed algorithm's extensive search.

```
repeat
    if there is a skipped block then
        Fly to this block
        Check palm
        if palm is severely infested then
            Call for help
            Block state = severe
        else if the palm is mildly infested then
            Treat palm
            Block state = treated
until all skipped blocks are detected
```

missions:

- The number of UAVs was logarithmically increased to the power of 2 to illustrate the performance of the algorithms in six different fleet scales: $2\hat{k}$, $k = 2, ...7$ UAVs.

- The number of infested palms was similarly logarithmically increased to the power of 2 at a larger scales, where 12 values for the number of infested palms was considered: $2\hat{k}$, $k = 1, ...12$ infested palms.

The performance of UTARB was evaluated against a total of three benchmark algorithms (auction-based, max-sum, and LIAM), as well as OTA which was used as a baseline. The total number of developed scenarios was 288: number of heuristics (4) $\times$ number of values for the number of infestations (12) $\times$ number of values for the number of UAVs (6). Further simulations were carried out with the proposed algorithm at a larger plantation scale to include 72 additional simulations, with a total of 360 scenarios. Each scenario was executed 10 times to reduce the variability of the performance metrics.

The OTA scheme is used as a baseline performance measure, which simply allocates UAVs to search the nearest block in the plantation area that has yet to be explored. If an infested palm is discovered, the UAV treats the palm and then resume its search for more infestations using the same strategy [27].

A typical auction-based coordination strategy commands all UAVs to search the plantation area and report a list of infestations found and their locations at the earliest upcoming auction event. A cost function for each infested palm, as well as the UAV's own ongoing tasks, is evaluated to place a bid. Infested palms are then assigned to the winning bidder by the auctioneer UAV. Once all infested palms are assigned to bid winners, the UAVs commence treating the infested palms and continue to do so until the next upcoming auction event occurs or the assigned treat tasks are completed. Once the tasks are completed and unchecked blocks still remain in the plantation area, the UAV will return to detect infestations until the next auction event occurs [76, 77].

The third benchmark algorithm, max-sum coordination, is an alternative form of the auction-based strategy. The algorithm performs a detect cycle similar to that of the auction algorithm. For the treat cycle, the agents perform a specific repetition of a non-greedy distribution algorithm. This algorithm uses factoring to allocate all the infested palms between the agents in an optimal distribution. The algorithm also depends on specific costs such as those associated with the battery, capacity, and infestation level factors. After repeating the max-sum algorithm, each infested palm is reassigned if new optimal agents are available to treat these infested palms. As with the auction algorithm, the agents then continue to conduct their new assignments until the next round of max-sum repetitions, or if no infested palm assignments remain, each agent is restored as a detect agent if the search area has not been completely explored.

The last benchmark algorithm is LIAM, a recently proposed heuristic that is inspired by the behaviour of locusts. The swarm behaviour of locusts is well-known as millions can gather to form huge swarms, while also being able to exist in solitary. This role-changing behaviour presents a biological example of adaptive control in response to internal and external stimuli [27]. LIAM was developed to mimic this behaviour and exploit the role changing property of locusts [17]. This is similar to the behaviour of agents in UTARB, where roles adapt to the environment. In the original paper, searcher UAVs start off as scout UAVs that follow a random search strategy at low speed and short distances. As all search areas are assigned, scouts change their roles to eagle UAVs where a guided search strategy is adopted at a medium speed for medium distances. Standby UAVs are not involved in the search process, however they are available for on-demand rescue with intermittent flights at high speed for long distances.The original LIAM was proposed for search and rescue mission and was thus adapted to DAT missions to comparatively assess the performance of UTARB. This is possible since LIAM is not limited to search and rescue missions and can be applied to other missions that involve dynamic allocation of two groups of task in multi-agent systems.

Table 1: Parameter settings of the evaluation environment.

Parameters	Settings
Plantation area size	100×100 regions 100 blocks per region 8×8 meters block size
Hotspots	10; radius: 200 meters, DOF: 2.5
Plane maximum speed	40 miles/hour
Block detect power consumption penalty	5 units of power
Block detect time penalty	10 seconds
Palm treatment power consumption penalty	10 units of power
Palm treatment time penalty	60 seconds
Idle power consumption	1 unit of power/300 milliseconds
Standard power consumption	1 unit of power/100 milliseconds

Two performance metrics were measured for each scenarios: net throughput and algorithm running time. The net throughput is used to measure the total number of detected and treated palms in the simulations. The second metric, algorithm running time, measures the time from when the simulation starts until its end. The latter measure is used to indicate the algorithm's complexity. The standard deviation for both metrics was also calculated to indicate the extent of the deviation across the ten trials for each of the 360 scenarios.

The simulated plantation area is comprised of a number of infested palms, where the locations, infestations level, and scattering were randomised to best simulate representative samples in all the experiments. The location of the infested palms were randomly generated using multivariate normal distributions that simulate hotspots of a specified radius. These hotspots showcase infested palms that are clustered together and are more likely to support the spread of RPW infestation. Several variables were generated and stored as test scenarios to ensure the repeatability of the testing process. This includes: number of infested palms, infestations level, palm locations, and the number of UAVs. At the start of each simulation, the initial locations of the UAVs is randomly generated to introduce variability and imitate real-world scenarios.

In the simulation, the plantation area was modelled as an area of size 100 $\times$ 100 regions (with 8 $\times$ 8 meter blocks). The maximum UAV speed was 40 miles/hour. The power consumption was uniform across all UAVs and power is assumed infinite for all. However, penalties were applied every time a block was explored (5 units of power) and a palm was treated (10 units of power). Time penalties of 10 and 60 seconds were also assumed when exploring a block and treating an infested palm, respectively. The power consumption during idle time and standard operational time was 1 unit of power/300 milliseconds and 1 unit of power/100 milliseconds. Hotspots are randomised locations simulated with a specific radius using a multivariate normal distribution, with a total of ten 200-meter-radius hotspots. Table 1 displays the parameter settings of the simulation environment that are utilised by UTARB and the benchmark algorithms (OTA, auction-based, max-sum, and LIAM). Further simulations were carried out for UTARB in a plantation area that was modelled as an area of size 1000 $\times$ 1000 regions with 8 $\times$ 8 meter blocks. The rest of the environment's parameters were maintained as presented in Table 1.

The MASPlanes multi-UAV simulator sustains defaults values for the auction-based and max-sum algorithms which were suited to the scenarios simulated in this paper. In the auction-based algorithm, auctions were conducted every 0.5 seconds and bids placed by UAVs are dependent on the auctioned task's cost. For the max-sum algorithms, the number of max-sum iterations to the point of reaching a decision was set at 9 iterations. The number of iterations between max-sum cycles was set at 10 iterations. Coordinations between the UAVs in the max-sum algorithm was via workload auctions. LIAM maintains several roles to mimic the behaviour of locusts: Scouts, Eagles, and Standby UAVs. LIAM introduces penalties for conversion from one to another, primarily a loss of approximately 44% of the battery capacity to convert a Scout UAV to Eagle UAV and approximately 8% to convert an Eagle UAV to Standby UAV. These roles and their parameters are described in the original paper and were utilised for the simulations [17].

For each scenario, the experiment was repeated ten times. The results were then averaged and presented in lin-log graphs defined by a logarithmic base-2 scale, with the number of UAVs on the x-axis, and with a linear scale for the performance measures on the y-axis (net throughput and algorithm's running time). Lin-log graphs employing a logarithmic axis allow for simultaneous comparisons of data points drawn from a wide range of UAVs (4-128). The standard deviations was also collected for the ten trials to present the spread out from the computed average.

7. Results and Discussion

The performance of UTARB is analysed in this section on DAT missions against four benchmark algorithms: OTA strategy, auction algorithm, max-sum coordination, and LIAM. The following subsections examine the performance of the four algorithms considering the two performance metrics: net throughput and algorithm running time. The analysis is conducted separately for each of the metrics.

7.1. Net Throughput

The net throughput of the system was computed by collecting the total number of detected and treated palms in each scenario. The percentage of detected and treated palms is plotted against the number of UAVs as the number of infestations is increased exponentially from 2 to 4,096 infestations (see Figure 5). The different scenarios are presented using radar graphs, where the axis and scale of 0 to 100% represent the net throughput, and the UAV fleet size (4-128) is shown on a separate axis. Figure 5 shows 12 subfigures, each of which displaying the net throughput for the UTARB, OTA, auction, max-sum, and LIAM algorithms. Because of the steep increase in the runtime performance of the auction and max-sum strategies, the algorithms halted at an early stage (see subfigures 5i -5l). The standard deviation for the 10 trials collected for each algorithm was also computed and presented in Table 2.

With only two infested palms in the first scenario (see subfigures 5a), all five algorithms achieved a net throughput of 100% as different UAV fleet sizes were deployed (i.e., all infested palms were detected and treated for infestation). A similar performance can be observed with only four infestations (see subfigure 5a); however, max-sum begins to lag behind, as it was only able to discover three of the four infested palms. As more infestations are spread throughout the plantation, the superior performance of the proposed UTARB algorithm becomes apparent. In all scenarios, UTARB was able to detect and treat all infested palms within the mission parameters at increasing fleet sizes and number of infested palms. Of the four benchmarks, the performance of LIAM was closest to that of UTARB with infestations spreading to no more than 512 palms (see subfigure 5i). As the number of infested palms increases to 1,024 the performance of LIAM falters with only

4 UAVs in its fleet (see subfigure 5j). The performance continues to decline with 2,048 and 4,096 UAVs and more UAVs (see subfigures 5k and 5l).

Although the two benchmark algorithms, auction and max-sum, achieved almost identical performances, max-sum proved to be less robust to increasing number of UAVs in a fleet and number of infestations (see subfigures 5i -5l). At 1,024 infestations, the performance of the auction-based algorithm begins to deteriorate as the fleet size increase to 128 UAVs. The algorithm progressively worsens as the number of infestations increases to 2,048 and 4,096 and halts with even fewer agents: 32 UAVs. The performance of max-sum coordination proved to be even poorer, as it halted earlier than the auction algorithm at only 512 infested palms in the plantation area and 32 UAVs. The algorithm failed to produce more results in subsequent scenarios as well, with fewer UAVs in the fleet. The deterioration of the algorithms' performances is likely because of the large number of iterations that are performed by the auction and max-sum algorithms. This, of course, left the system unable to handle the increasing number of tasks.

At only a few infestations, the OTA strategy and LIAM perform well compared to the other two benchmark algorithms. Whereas the auction-based and max-sum coordinations halted as infestations grew in number, OTA and LIAM were able to progress and detect infestations. However, the percentage of treated palms clearly diminishes when only a few UAVs are deployed for most scenarios for OTA (see subfigures 5d-5l). This was also the case for the auction and max-sum algorithms, but to a relatively lower extent, as the two algorithms produced better net throughput than OTA. LIAM, on the other hand, produced significantly higher net throughput compared to the other benchmarks with reasonable runtime performance. At 2,048 and 4,096 infestations the runtime performance of LIAM lies in contrast to that of OTA, where the increasing number of UAVs improved LIAM's performance. Nevertheless, unlike the four benchmark strategies, the proposed UTARB algorithm's net throughput was not affected by the increase in the number of UAVs. This demonstrates its ability to conduct the mission efficiently and economically with fewer UAVs.

Additional simulations were carried out with UTARB to inspect its performance in a larger plantation area (1000×1000 regions) with logarithmically increasing UAV fleets and infestations. Similar to what was observed in the smaller area, the UTARB algorithm was able to detect and treat all infestations in this larger plantation. This shows that the performance of UTARB is scalable to larger plantations, while still maintaining its economic advantage.

7.2. Runtime Performance

The runtime performances of the UTARB, OTA, auction, max-sum, and LIAM algorithms were recorded for each of the simulated scenarios. The results of the average runtime performances and standard deviation are shown in Table 3 and illustrated in Figure 6 as lin-log graphs as the number of infestations grew from 2 to 4,096 infested palms against the number of UAVs. The dashed lines in the figures represent values and lines that extend sharply beyond the values displayed in the vertical axis. Therefore, Table 3 is included to report these values. As previously indicated in the previous section and Figure 5, the auction-based and max-sum algorithms halted in scenarios with a large number of infested palms because of the large computational overhead.

The results show that UTARB outperformed the other benchmark algorithms in this performance measure as well. This is especially the case with larger infestations and a larger number of UAVs (see subfigures 6j-6l). It is important to note that the performance of the proposed algorithm and OTA was similar in this metric when only a few UAVs were deployed at lower infestation levels (up to 16 infested palms). However, the runtime performance of the OTA algorithm steadily increased with the number of UAVs in the fleet and number of infested palms. The number of deployed UAVs in the scenarios had minimal impact on the runtime performance of UTARB, where only slight variations occurred across the scenarios. This marginal impact on the performance of UTARB at the different levels of infestation and fleet size demonstrates the algorithm's capability of economically completing the DAT mission with fewer UAVs.

LIAM's running time was relatively close to that of UTARB and OTA at lower levels of infestations. Similar to what was observed for the OTA algorithm, the running time of LIAM steadily grows as the number of infestations was increased from 2 to 4,096 infested palms. At lower levels of infestations (2-8 infested palms), the running time of OTA is almost twice as fast as that of LIAM. However, this number starts to significantly decrease as the number of infestations grow with LIAM outperforming the OTA algorithm. Similar to UTARB, the variations in running time for LIAM as the number of UAVs was increased was relatively small. However, as the running time of the UTARB algorithm increased those for LIAM begin to decrease as the fleet size grows for higher levels of infestations. This shows that while

UTARB is able to sustain its running time with smaller and larger fleets, LIAM's performance is significantly better with larger fleets of UAVs.

The auction and max-sum strategy required considerably more time to detect infestation in the DAT mission than did the proposed algorithm and the baseline. The runtime of the auction and max-sum algorithms significantly increased, taking up to 16 and 20 hours for 4,096 infestations, respectively. For both strategies, the system halted before showing meaningful results starting at 512 infestations and above (see subfigures 6i-6l). In subfigure 6g-6l, the lines representing the runtime performance of the max-sum algorithm extend far beyond the x-axis because of the large disparity between its values and those of the other algorithms as the number of infestations increased. This was also the case for the auction algorithm as the number of UAVs was increased in the scenarios. These large variations for both algorithms are likely because of the amplified complexity as more UAVs were introduced.

The running time of the UTARB algorithm was further examined in a larger plantation area of 1000×1000 regions (with 8×8 meter blocks) as the number of UAVs were increased along with the number of infested palms (see Figure 7). Table 4 shows UTARB's running time for a small and a large plantation area. The findings show that UTARB's running time in the bigger area is understandably larger than that of the smaller area. UTARB's running time as the number of UAVs increases almost doubles for the larger area and variations are slightly bigger than those observed in the smaller area. In fact, in a larger area, UTARB appears to perform better with fewer UAVs. This finding continues to demonstrate the economic value of UTARB as it is able to complete the DAT mission with fewer UAVs.

8. Conclusions

In DAT missions, the deployment of multiple UAVs requires the proper allocation of tasks to efficiently detect and treat pest infestations. The MUTA problem is addressed in this paper with UTARB, a bacteria-inspired problem-based heuristic. UTARB's computational parameters are simple and are measured by each UAV locally. This gives each UAV the capability of making independent task allocation decisions to ensure autonomy and low running times. A simulator was built to thoroughly assess the performance of the proposed bio-inspired heuristic against three well-established benchmark algorithms and a recently proposed problem-dependent bio-inspired algorithm. The experimental findings demonstrate that using UTARB for task allocation in DAT missions considerably increased the percentage of detected and treated RPW infested palms and reduced the overall runtime complexity under different scales of fleet size and number of infestations. The results also highlighted the economic value of UTARB, as fewer UAVs were required to quickly detect and treat infestations and halt their spread.

Although UTARB was introduced within the context of DAT missions, it is by no means limited to this application domain. The algorithm can potentially be applied in missions such as search-and-monitor missions for goods and items in factories and warehouse and survey-and-treatment missions for livestock, crops, and forest protection. The work in this paper opens up several interesting research directions. The proposed UTARB algorithm outperformed conventional algorithms, and we plan to assess it behaviours when compared to well-established bio-inspired algorithms. Additionally, only independent tasks were considered; we intend to explore the task allocation problem with dependent tasks and workflows. Furthermore, this work was done as a part of a national project aimed at combating RPW infestation in Saudi Arabia and thus will be deployed in a field setting to assess the performance of the algorithm.

Acknowledgements

This work was supported by Saudi Aramco, under the "Saudi Aramco Ibn Khaldun Fellowship for Saudi Women," in partnership with the Center for Clean Water and Clean Energy at MIT, and the International Scientific Partnership Program ISPP at King Saud University.

Declarations of interest

The authors declare no competing interest.

References

[1] K. Tofailli, et al., The early detection of red palm weevil: a new method, Acta horticulturae 882 (2010) 441–449.

[2] Ó. Dembilio, J. A. Jaques, Biology and management of red palm weevil, in: Sustainable Pest Management in Date Palm: Current Status and Emerging Challenges, Springer, 2015, pp. 13–36.

[3] A. Muhammad, Y. Fang, Y. Hou, Z. Shi, The gut entomotype of red palm weevil rhynchophorus ferrugineus olivier (coleoptera: Dryophthoridae) and their effect on host nutrition metabolism, Frontiers in microbiology 8 (2017) 2291.

[4] A. Hetzroni, V. Soroker, Y. Cohen, Toward practical acoustic red palm weevil detection, Computers and electronics in agriculture 124 (2016) 100–106.

[5] V. Soroker, P. Suma, A. La Pergola, V. N. Llopis, S. Vacas, Y. Cohen, Y. Cohen, V. Alchanatis, P. Milonas, O. Golomb, et al., Surveillance techniques and detection methods for rhynchophorus ferrugineus and paysandisia archon, Handbook of Major Palm Pests: Biology and Management (2017) 209–232.

[6] P. Bilski, P. Bobiński, A. Krajewski, P. Witomski, Detection of wood boring insects? larvae based on the acoustic signal analysis and the artificial intelligence algorithm, Archives of Acoustics 42 (1) (2017) 61–70.

[7] S. Baena, J. Moat, O. Whaley, D. S. Boyd, Identifying species from the air: Uavs and the very high resolution challenge for plant conservation, PloS one 12 (11) (2017) e0188714.

[8] R. Chen, T. Chu, J. A. Landivar, C. Yang, M. M. Maeda, Monitoring cotton (gossypium hirsutum l.) germination using ultrahigh-resolution uas images, Precision Agriculture 19 (1) (2018) 161–177.

[9] R. Näsi, E. Honkavaara, M. Blomqvist, P. Lyytikäinen-Saarenmaa, T. Hakala, N. Viljanen, T. Kantola, M. Holopainen, Remote sensing of bark beetle damage in urban forests at individual tree level using a novel hyperspectral camera from uav and aircraft, Urban Forestry & Urban Greening 30 (2018) 72–83.

[10] W. Xiong, D. W. Ho, J. Cao, W. X. Zheng, Backstepping approach to a class of hierarchical multi-agent systems with communication disturbance, IET Control Theory & Applications 10 (9) (2016) 981–988.

[11] F. Ye, W. Zhang, L. Ou, G. Zhang, Optimal disturbance rejection controllers design for synchronised output regulation of time-delayed multi-

agent systems, IET Control Theory & Applications 11 (7) (2017) 1053–1062.

[12] E. Nunes, M. Manner, H. Mitiche, M. Gini, A taxonomy for task allocation problems with temporal and ordering constraints, Robotics and Autonomous Systems 90 (2017) 55–70.

[13] J. Y. Leung, Handbook of scheduling: algorithms, models, and performance analysis, CRC Press, 2004.

[14] M. Pujol-Gonzalez, J. Cerquides, A. Farinelli, P. Meseguer, J. A. Rodriguez-Aguilar, Efficient inter-team task allocation in robocup rescue, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 413–421.

[15] B. Niu, F. T. Chan, T. Xie, Y. Liu, Guided chemotaxis-based bacterial colony algorithm for three-echelon supply chain optimisation, International Journal of Computer Integrated Manufacturing 30 (2-3) (2017) 305–319.

[16] J. Tang, K. Zhu, H. Guo, C. Gong, C. Liao, S. Zhang, Using auction-based task allocation scheme for simulation optimization of search and rescue in disaster relief, Simulation Modelling Practice and Theory 82 (2018) 132–146.

[17] H. A. Kurdi, E. Aloboud, M. Alalwan, S. Alhassan, E. Alotaibi, G. Bautista, J. P. How, Autonomous task allocation for multi-uav systems based on the locust elastic behavior, Applied Soft Computing.

[18] F. R. Kschischang, B. J. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, IEEE Transactions on information theory 47 (2) (2001) 498–519.

[19] M. Pujol-Gonzalez, J. Cerquides, A. Farinelli, P. Meseguer, J. A. Rodriguez-Aguilar, Binary max-sum for multi-team task allocation in robocup rescue.

[20] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, N. R. Jennings, Decentralized coordination in robocup rescue, The Computer Journal 53 (9) (2010) 1447–1461.

[21] A. Corrêa, Binary max-sum for clustering-based task allocation in the rmasbench platform, in: Evolutionary Computation (CEC), 2016 IEEE Congress on, IEEE, 2016, pp. 1046–1053.

[22] E. Schneider, E. I. Sklar, S. Parsons, A. T. Özgelen, Auction-based task allocation for multi-robot teams in dynamic environments, in: Conference Towards Autonomous Robotic Systems, Springer, 2015, pp. 246–257.

[23] L. Wang, M. Liu, M. Q.-H. Meng, A hierarchical auction-based mechanism for real-time resource allocation in cloud robotic systems, IEEE transactions on cybernetics 47 (2) (2017) 473–484.

[24] Y. Ma, B. Li, Y. Zhang, J. Zhu, Efficient auction mechanism with group price for resource allocation in clouds, in: Advanced Cloud and Big Data (CBD), 2014 Second International Conference on, IEEE, 2014, pp. 85–92.

[25] P. Li, H. Duan, Bio-inspired computation algorithms, in: Bio-inspired Computation in Unmanned Aerial Vehicles, Springer, 2014, pp. 35–69.

[26] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26 (1) (1996) 29–41.

[27] H. Kurdi, J. How, G. Bautista, Bio-inspired algorithm for task allocation in multi-uav search and rescue missions, in: AIAA Guidance, Navigation, and Control Conference, 2016, p. 1377.

[28] Y. Liu, K. Passino, Biomimicry of social foraging bacteria for distributed optimization: models, principles, and emergent behaviors, Journal of optimization theory and applications 115 (3) (2002) 603–628.

[29] S. D. Muller, J. Marchetto, S. Airaghi, P. Kournoutsakos, Optimization based on bacterial chemotaxis, IEEE transactions on Evolutionary Computation 6 (1) (2002) 16–29.

[30] X. Liu, L. Huang, X. Chang, The bacteria foraging algorithm for global optimization based on pheromone, in: Service Systems and Service Management (ICSSSM), 2015 12th International Conference on, IEEE, 2015, pp. 1–7.

[31] S. Devi, M. Geethanjali, Application of modified bacterial foraging optimization algorithm for optimal placement and sizing of distributed generation, Expert Systems with Applications 41 (6) (2014) 2772–2781.

[32] S. Abd-Elazim, E. Ali, A hybrid particle swarm optimization and bacterial foraging for power system stability enhancement, Complexity 21 (2) (2015) 245–255.

[33] A. Nasir, M. O. Tokhi, Novel metaheuristic hybrid spiral-dynamic bacteria-chemotaxis algorithms for global optimisation, Applied Soft Computing 27 (2015) 357–375.

[34] B. Turanoğlu, G. Akkaya, A new hybrid heuristic algorithm based on bacterial foraging optimization for the dynamic facility layout problem, Expert Systems with Applications 98 (2018) 93–104.

[35] W.-w. Li, H. Wang, Z.-j. Zou, J.-x. QIAN, Function optimization method based on bacterial colony chemotaxis, Journal of Circuits and Systems 10 (1) (2005) 58–63.

[36] B. Xing, W.-J. Gao, Bacteria inspired algorithms, in: Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms, Springer, 2014, pp. 21–38.

[37] C. Anandaraman, A. V. M. Sankar, R. Natarajan, A new evolutionary algorithm based on bacterial evolution and its application for scheduling a flexible manufacturing system, Jurnal Teknik Industri 14 (1) (2012) 1–12.

[38] P. Cortés, J. M. García, J. Muñuzuri, L. Onieva, Viral systems: A new bio-inspired optimisation approach, Computers & Operations Research 35 (9) (2008) 2840–2860.

[39] J. J. Bartholdi III, L. K. Platzman, Heuristics based on spacefilling curves for combinatorial problems in euclidean space, Management Science 34 (3) (1988) 291–305.

[40] S. Al-Megren, H. Kurdi, M. F. Aldaood, A multi-uav task allocation algorithm combatting red palm weevil infestation, Procedia Computer Science 141 (2018) 88–95.

[41] A. Farinelli, A. Rogers, A. Petcu, N. R. Jennings, Decentralised coordination of low-power embedded devices using the max-sum algorithm, in: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2, International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 639–646.

[42] F. M. Delle Fave, A. Rogers, Z. Xu, S. Sukkarieh, N. R. Jennings, Deploying the max-sum algorithm for decentralised coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, 2012, pp. 469–476.

[43] F. M. Delle Fave, A. Farinelli, A. Rogers, N. Jennings, A methodology for deploying the max-sum algorithm and a case study on unmanned aerial vehicles.

[44] S. D. Ramchurn, T. D. Huynh, F. Wu, Y. Ikuno, J. Flann, L. Moreau, J. E. Fischer, W. Jiang, T. Rodden, E. Simpson, et al., A disaster response system based on human-agent collectives, Journal of Artificial Intelligence Research 57 (2016) 661–708.

[45] H.-L. Choi, L. Brunet, J. P. How, Consensus-based decentralized auctions for robust task allocation, IEEE transactions on robotics 25 (4) (2009) 912–926.

[46] M. Elango, S. Nachiappan, M. K. Tiwari, Balancing task allocation in multi-robot systems using k-means clustering and auction based mechanisms, Expert Systems with Applications 38 (6) (2011) 6486–6491.

[47] S. Binitha, S. S. Sathya, et al., A survey of bio inspired optimization algorithms, International Journal of Soft Computing and Engineering 2 (2) (2012) 137–151.

[48] A. Jevtic, A. Gutiérrez, D. Andina, M. Jamshidi, Distributed bees algorithm for task allocation in swarm of robots, IEEE Systems Journal 6 (2) (2012) 296–304.

[49] M. Akbari, H. Rashidi, A multi-objectives scheduling algorithm based on cuckoo optimization for task allocation problem at compile time in heterogeneous systems, Expert Systems with Applications 60 (2016) 234–248.

[50] T. Zheng, L. Yang, Optimal ant colony algorithm based multi-robot task allocation and processing sequence scheduling, in: Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on, IEEE, 2008, pp. 5693–5698.

[51] L. Wang, Z. Wang, S. Hu, L. Liu, Ant colony optimization for task allocation in multi-agent systems, China Communications 10 (3) (2013) 125–132.

[52] H. Wu, H. Li, R. Xiao, J. Liu, Modeling and simulation of dynamic ant colony?s labor division for task allocation of uav swarm, Physica A: Statistical Mechanics and its Applications 491 (2018) 127–141.

[53] M. Turduev, M. Kirtay, P. Sousa, V. Gazi, L. Marques, Chemical concentration map building through bacterial foraging optimization based search algorithm by mobile robots, in: Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on, IEEE, 2010, pp. 3242–3249.

[54] M. Turduev, G. Cabrita, M. Kırtay, V. Gazi, L. Marques, Experimental studies on chemical concentration map building by a multi-robot system using bio-inspired algorithms, Autonomous agents and multi-agent systems 28 (1) (2014) 72–100.

[55] S. Sharma, C. Sur, A. Shukla, R. Tiwari, Multi robot path planning for known and unknown target using bacteria foraging algorithm, in: International Conference on Swarm, Evolutionary, and Memetic Computing, Springer, 2014, pp. 674–685.

[56] M. A. Hossain, I. Ferdous, Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique, Robotics and Autonomous Systems 64 (2015) 137–141.

[57] B. Yang, Y. Ding, Y. Jin, K. Hao, Self-organized swarm robot for target search and trapping inspired by bacterial chemotaxis, Robotics and Autonomous Systems 72 (2015) 83–92.

[58] J. Cao, M. Li, H. Wang, L. Huang, Y. Zhao, An improved bacterial foraging algorithm with cooperative learning for eradicating cancer cells

using nanorobots, in: Robotics and Biomimetics (ROBIO), 2016 IEEE International Conference on, IEEE, 2016, pp. 1141–1146.

[59] F. Dressler, O. B. Akan, A survey on bio-inspired networking, Computer Networks 54 (6) (2010) 881–900.

[60] R. S. Xavier, N. Omar, L. N. de Castro, Bacterial colony: Information processing and computational behavior, in: Nature and biologically inspired computing (NaBIC), 2011 Third World Congress on, IEEE, 2011, pp. 439–443.

[61] H. Duan, P. Li, Bio-inspired computation in unmanned aerial vehicles, Springer, 2014.

[62] A. Afshar, F. Massoumi, A. Afshar, M. A. Mariño, State of the art review of ant colony optimization applications in water resource management, Water resources management 29 (11) (2015) 3891–3904.

[63] M. Iqbal, M. Naeem, A. Ahmed, M. Awais, A. Anpalagan, A. Ahmad, Swarm intelligence based resource management for cooperative cognitive radio network in smart hospitals, Wireless Personal Communications 98 (1) (2018) 571–592.

[64] M. F. Allawi, O. Jaafar, M. Ehteram, F. M. Hamzah, A. El-Shafie, Synchronizing artificial intelligence models for operating the dam and reservoir system, Water Resources Management (2018) 1–17.

[65] Y. Chen, M. Hu, A swarm intelligence based distributed decision approach for transactive operation of networked building clusters, Energy and Buildings 169 (2018) 172–184.

[66] V. E. Karpov, V. B. Tarassov, Synergetic artificial intelligence and social robotics, in: International Conference on Intelligent Information Technologies for Industry, Springer, 2017, pp. 3–15.

[67] D. Hein, A. Hentschel, T. A. Runkler, S. Udluft, Particle swarm optimization for model predictive control in reinforcement learning environments, in: Critical Developments and Applications of Swarm Intelligence, IGI Global, 2018, pp. 401–427.

[68] I. BoussaïD, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information Sciences 237 (2013) 82–117.

[69] K. M. Passino, Bacterial foraging optimization, in: Innovations and Developments of Swarm Intelligence Applications, IGI Global, 2012, pp. 219–234.

[70] M. A. Munoz, S. K. Halgamuge, W. Alfonso, E. F. Caicedo, Simplifying the bacteria foraging optimization algorithm, in: Evolutionary Computation (CEC), 2010 IEEE Congress on, IEEE, 2010, pp. 1–7.

[71] Q.-Y. Zhao, M. Li, J. Luo, Y. Li, L. Dou, A nanorobot swarming algorithm based on bacteria foraging optimization to eradicate cancer cells, in: Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on, IEEE, 2014, pp. 2599–2604.

[72] D. B. Kearns, A field guide to bacterial swarming motility, Nature Reviews Microbiology 8 (9) (2010) 634.

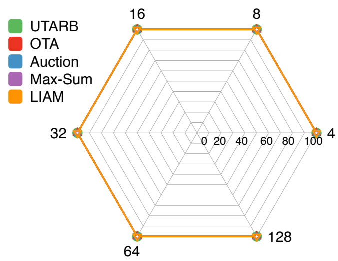
[73] H. A. Kurdi, Personal mobile grids with a honeybee inspired resource scheduler, Ph.D. thesis, Brunel University School of Engineering and Design PhD Theses (2010).

[74] R. G. Askin, C. R. Standridge, Modeling and analysis of manufacturing systems, Vol. 29, Wiley New York, 1993.

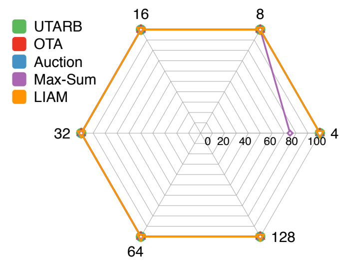
[75] M. Pujol-Gonzalez, J. Cerquides, P. Meseguer, Mas-planes: a multi-agent simulation environment to investigate decentralised coordination for teams of uavs, in: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1695–1696.

[76] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, V. Segui, A combinatorial auction framework for decentralised task allocation, in: Globecom Workshops (GC Wkshps), 2014, IEEE, 2014, pp. 1445–1450.

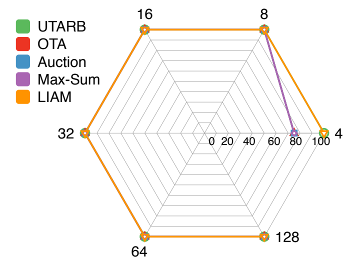
[77] L. Johnson, H.-L. Choi, J. P. How, The hybrid information and plan consensus algorithm with imperfect situational awareness, in: Distributed Autonomous Robotic Systems, Springer, 2016, pp. 221–233.



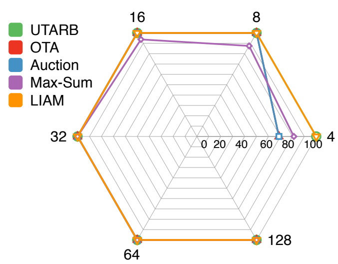
(a) 2 infested palms.



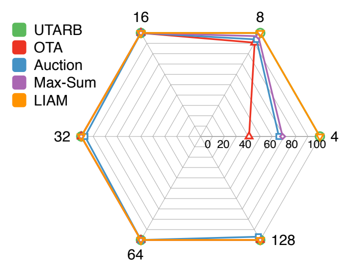
(b) 4 infested palms.



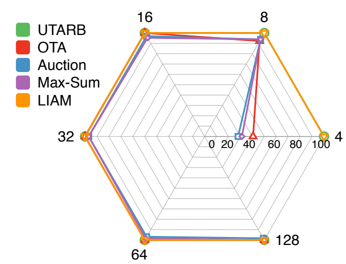
(c) 8 infested palms.



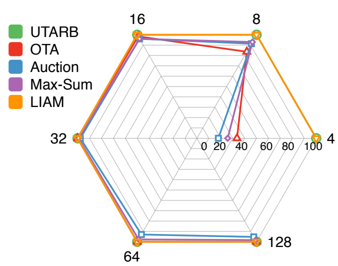
(d) 16 infested palms.



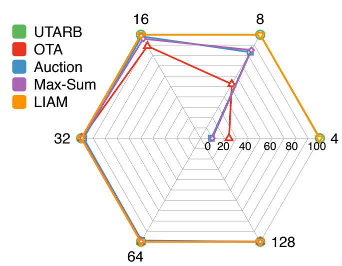
(e) 32 infested palms.



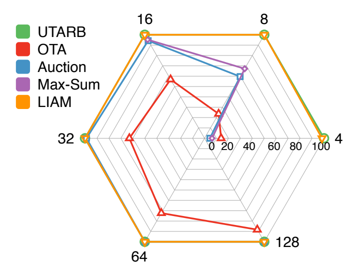
(f) 64 infested palms.



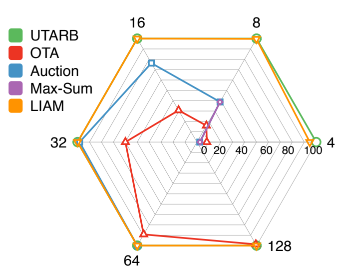
(g) 128 infested palms.



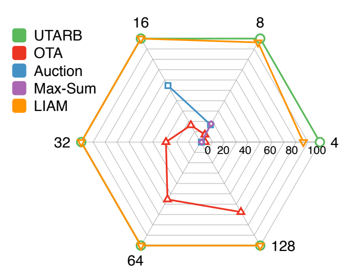
(h) 256 infested palms.



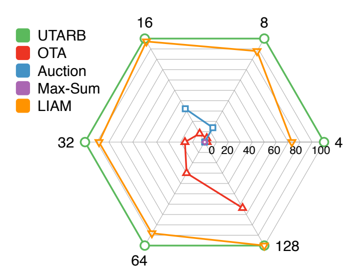
(i) 512 infested palms.



(j) 1,024 infested palms.



(k) 2,048 infested palms.



(l) 4,096 infested palms.

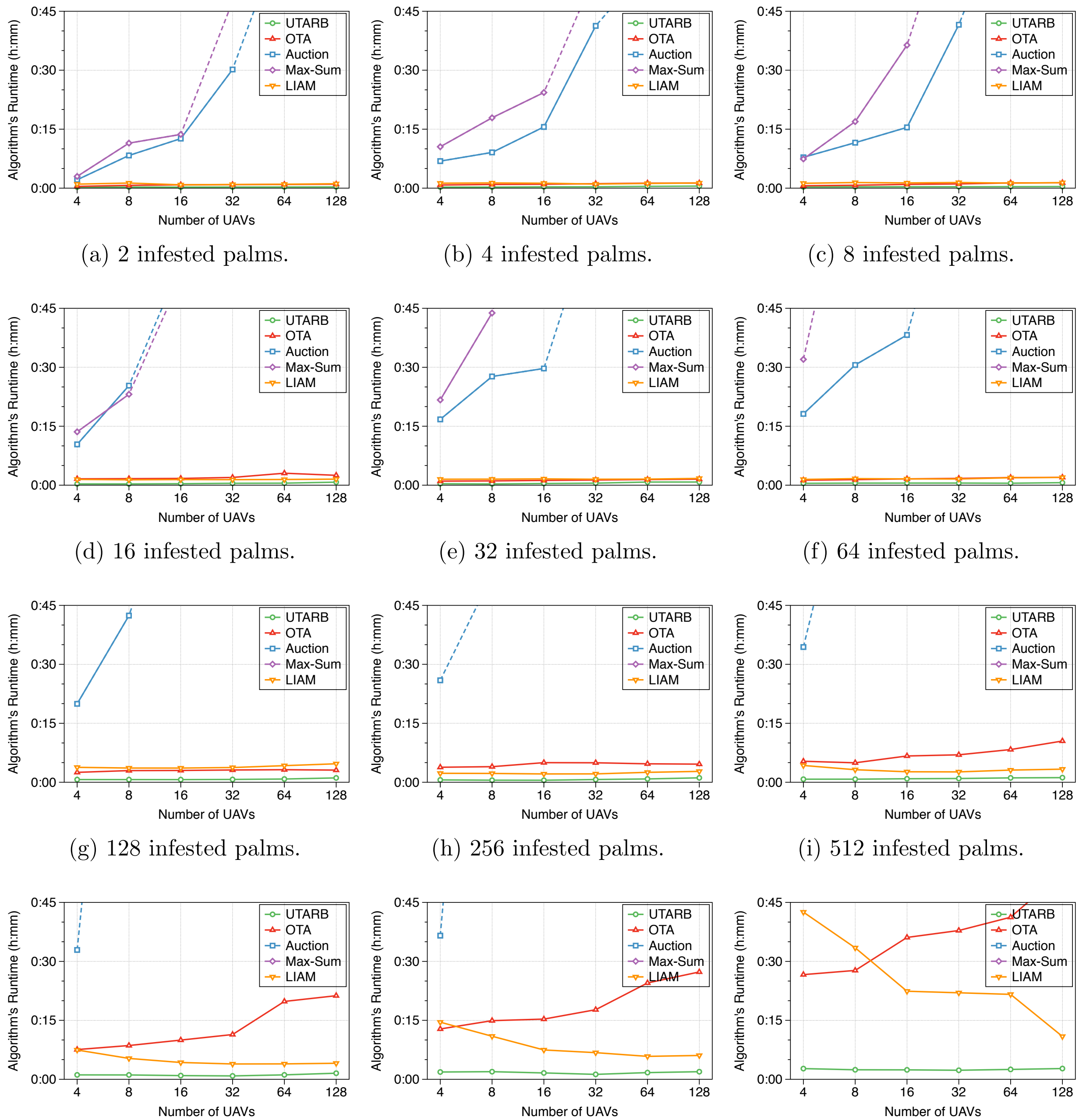
Figure 5: Net throughput as the number of infested palms increases from 2 to 4,096 in the plantation area.

Table 2: Net throughput standard deviation for total number of detected and treated palms (presented as a percentage %) trials as the number of infestations increases from 2 to 4,096 in the plantation area.

Infestations	UAVs	UTARB	OTA	Auction	Max-Sum	LIAM
2	4	0.00	0.00	0.00	0.00	0.00
	8	0.00	0.00	0.00	0.00	0.00
	16	0.00	0.00	0.00	0.00	0.00
	32	0.00	0.00	0.00	0.00	0.00
	64	0.00	0.00	0.00	0.00	0.00
	128	0.00	0.00	0.00	0.00	0.00
4	4	0.00	0.00	0.00	3.14	0.00
	8	0.00	0.00	0.00	0.00	0.00
	16	0.00	0.00	0.00	0.00	0.00
	32	0.00	0.00	0.00	0.00	0.00
	64	0.00	0.00	0.00	0.00	0.00
	128	0.00	0.00	0.00	0.00	0.00
8	4	0.00	3.70	0.00	0.00	0.00
	8	0.00	0.00	0.00	0.00	0.00
	16	0.00	0.00	0.00	0.00	0.00
	32	0.00	0.00	0.00	0.00	0.00
	64	0.00	0.00	0.00	0.00	0.00
	128	0.00	0.00	0.00	0.00	0.00
16	4	0.00	3.21	4.85	5.43	0.00
	8	0.00	0.00	0.00	0.00	0.00
	16	0.00	0.00	0.00	0.00	0.00
	32	0.00	0.00	0.00	0.00	0.00
	64	0.00	0.00	0.00	0.00	0.00
	128	0.00	0.00	0.00	0.00	0.00
32	4	0.00	4.10	1.81	3.89	0.00
	8	0.00	1.21	2.99	2.29	0.00
	16	0.00	0.00	0.00	0.00	0.00
	32	0.00	0.00	0.98	0.00	0.00
	64	0.00	0.00	0.00	0.00	0.00
	128	0.00	0.00	2.41	0.00	0.00
64	4	0.00	5.66	4.34	5.96	0.00
	8	0.00	0.96	4.71	5.57	0.00
	16	0.00	0.00	2.05	2.21	0.00
	32	0.00	0.00	5.82	1.83	0.00
	64	0.00	0.00	3.32	5.61	0.00
	128	0.00	0.00	2.57	0.00	0.00
128	4	0.00	5.92	3.09	0.58	0.00
	8	0.00	1.15	3.27	1.99	0.00
	16	0.00	5.70	5.48	4.46	0.00
	32	0.00	0.00	0.96	4.04	0.00
	64	0.00	0.00	3.39	1.72	0.00
	128	0.00	0.00	1.73	2.26	0.00
256	4	0.00	3.20	2.18	2.35	1.00
	8	0.00	0.31	3.10	2.35	0.00
	16	0.00	1.60	1.97	4.94	0.00
	32	0.00	4.48	5.71	1.83	0.00
	64	0.00	3.03	2.16	0.38	0.00
	128	0.00	1.63	0.00	0.00	0.00
512	4	0.00	2.03	0.17	2.16	1.60
	8	0.00	4.84	3.48	0.60	0.00
	16	0.00	5.69	4.92	2.29	0.00
	32	0.00	4.96	2.14	/	0.00
	64	0.00	5.67	0.00	/	0.00
	128	0.00	4.70	0.00	/	0.50
1,024	4	0.00	2.89	4.67	1.92	2.10
	8	0.00	2.05	2.16	3.33	1.10
	16	0.00	2.38	5.47	/	0.00
	32	0.00	1.66	0.77	/	0.00
	64	0.00	5.89	0.00	/	0.00
	128	0.00	2.07	/	/	0.00
2,048	4	0.00	3.55	0.33	3.72	2.31
	8	0.00	2.61	1.47	2.19	1.20
	16	0.00	2.19	0.67	/	0.00
	32	0.00	3.34	/	/	0.00
	64	0.00	2.92	/	/	0.00
	128	0.00	3.18	/	/	0.00
4,096	4	0.00	2.29	3.22	0.72	2.80
	8	0.00	1.61	3.62	/	3.11
	16	0.00	2.21	2.25	/	0.90
	32	0.00	5.23	/	/	1.20
	64	0.00	1.42	/	/	1.31
	128	0.00	0.35	/	/	0.00

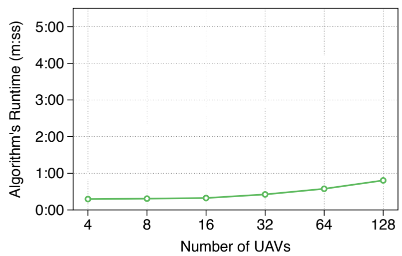
Table 3: Average runtime performance and standard deviation (h:mm:ss) as the number of infestations increases from 2 to 4,096 in the plantation area.

Infestations	UAVs	UTARB	OTA	Auction	Max-Sum	LIAM
2	4	0:00:15 (0:00:03)	0:00:26 (0:00:11)	0:02:07 (0:00:47)	0:02:58 (0:00:09)	0:01:01 (0:00:17)
	8	0:00:16 (0:00:02)	0:00:42 (0:00:15)	0:08:18 (0:01:43)	0:11:27 (0:01:43)	0:01:17 (0:00:17)
	16	0:00:15 (0:00:04)	0:00:50 (0:00:08)	0:12:36 (0:07:21)	0:13:40 (0:00:54)	0:00:50 (0:00:08)
	32	0:00:16 (0:00:13)	0:00:53 (0:00:12)	0:30:11 (0:10:34)	0:47:29 (0:03:40)	0:00:56 (0:00:19)
	64	0:00:17 (0:00:05)	0:00:56 (0:00:04)	1:06:08 (0:21:50)	1:29:06 (0:04:29)	0:00:59 (0:00:20)
	128	0:00:18 (0:00:08)	0:01:00 (0:00:05)	1:33:06 (0:11:40)	2:10:43 (0:17:17)	0:01:07 (0:00:17)
4	4	0:00:15 (0:00:03)	0:00:47 (0:00:10)	0:06:52 (0:02:09)	0:10:31 (0:01:21)	0:01:15 (0:00:17)
	8	0:00:15 (0:00:04)	0:00:56 (0:00:12)	0:09:05 (0:00:37)	0:17:53 (0:01:06)	0:01:19 (0:00:05)
	16	0:00:19 (0:00:03)	0:01:00 (0:00:05)	0:15:34 (0:06:27)	0:24:19 (0:02:53)	0:01:15 (0:00:11)
	32	0:00:20 (0:00:09)	0:01:07 (0:00:06)	0:41:16 (0:11:08)	0:54:39 (0:02:26)	0:01:02 (0:00:06)
	64	0:00:27 (0:00:07)	0:01:16 (0:00:08)	0:56:27 (0:11:52)	2:53:41 (0:09:27)	0:01:11 (0:00:16)
	128	0:00:31 (0:00:05)	0:01:17 (0:00:12)	2:32:58 (0:23:58)	4:06:43 (0:12:20)	0:01:18 (0:00:16)
8	4	0:00:18 (0:00:04)	0:00:36 (0:00:09)	0:07:50 (0:01:45)	0:07:25 (0:01:17)	0:01:14 (0:00:03)
	8	0:00:19 (0:00:06)	0:00:44 (0:00:06)	0:11:33 (0:03:46)	0:16:55 (0:00:41)	0:01:25 (0:00:12)
	16	0:00:20 (0:00:03)	0:00:57 (0:00:08)	0:15:27 (0:02:49)	0:36:22 (0:03:23)	0:01:19 (0:00:06)
	32	0:00:21 (0:00:02)	0:01:04 (0:00:10)	0:41:33 (0:06:54)	1:16:31 (0:05:40)	0:01:25 (0:00:16)
	64	0:00:22 (0:00:08)	0:01:19 (0:00:11)	1:14:47 (0:10:44)	2:29:53 (0:10:33)	0:01:17 (0:00:20)
	128	0:00:23 (0:00:04)	0:01:21 (0:00:19)	3:14:08 (0:11:06)	5:22:56 (0:22:23)	0:01:23 (0:00:10)
16	4	0:00:19 (0:00:05)	0:01:37 (0:00:11)	0:10:23 (0:01:13)	0:13:34 (0:02:48)	0:01:28 (0:00:08)
	8	0:00:19 (0:00:05)	0:01:39 (0:00:07)	0:25:20 (0:07:34)	0:23:07 (0:05:27)	0:01:22 (0:00:17)
	16	0:00:22 (0:00:02)	0:01:43 (0:00:04)	0:56:11 (0:10:09)	0:52:39 (0:04:19)	0:01:25 (0:00:11)
	32	0:00:29 (0:00:08)	0:01:58 (0:00:14)	1:06:20 (0:06:45)	1:38:41 (0:11:35)	0:01:24 (0:00:09)
	64	0:00:31 (0:00:03)	0:03:02 (0:00:14)	1:15:16 (0:20:27)	3:51:22 (0:28:46)	0:01:27 (0:00:06)
	128	0:00:46 (0:00:06)	0:02:30 (0:00:19)	2:35:49 (0:13:50)	6:37:45 (1:02:23)	0:01:32 (0:00:05)
32	4	0:00:21 (0:00:03)	0:01:01 (0:00:03)	0:16:44 (0:02:07)	0:21:41(0:01:30)	0:01:31 (0:00:08)
	8	0:00:18 (0:00:03)	0:01:05 (0:00:10)	0:27:39 (0:02:41)	0:43:48 (0:03:35)	0:01:32 (0:00:14)
	16	0:00:25 (0:00:06)	0:01:13 (0:00:11)	0:29:41 (0:06:27)	1:30:38 (0:05:20)	0:01:36 (0:00:12)
	32	0:00:31 (0:00:05)	0:01:19 (0:00:13)	1:10:33 (0:10:50)	3:07:49 (0:07:46)	0:01:30 (0:00:17)
	64	0:00:47 (0:00:04)	0:01:26 (0:00:04)	1:49:08 (0:10:44)	6:46:00 (0:10:42)	0:01:32 (0:00:15)
	128	0:00:48 (0:00:03)	0:01:32 (0:00:11)	3:37:57 (0:14:21)	10:43:39 (0:12:47)	0:01:43 (0:00:09)
64	4	0:00:28 (0:00:02)	0:01:15 (0:00:11)	0:18:07 (0:02:21)	0:32:00 (0:03:45)	0:01:28 (0:00:17)
	8	0:00:31 (0:00:04)	0:01:24 (0:00:12)	0:30:35 (0:01:34)	1:31:25 (0:08:16)	0:01:40 (0:00:08)
	16	0:00:32 (0:00:05)	0:01:36 (0:00:03)	0:38:13 (0:01:08)	2:48:14 (0:10:32)	0:01:36 (0:00:06)
	32	0:00:32 (0:00:06)	0:01:43 (0:00:13)	1:25:04 (0:10:33)	5:38:01 (0:13:48)	0:01:34 (0:00:10)
	64	0:00:30 (0:00:06)	0:01:56 (0:00:13)	3:17:19 (0:12:51)	10:44:42 (0:07:52)	0:01:53 (0:00:09)
	128	0:00:38 (0:00:04)	0:01:59 (0:00:06)	5:24:07 (0:11:36)	15:14:11 (0:20:06)	0:01:58 (0:00:16)
128	4	0:00:40 (0:00:04)	0:02:31 (0:00:11)	0:19:56 (0:01:40)	1:37:42 (0:10:25)	0:03:47 (0:00:12)
	8	0:00:40 (0:00:03)	0:02:57 (0:00:19)	0:42:24 (0:04:19)	3:24:23 (0:12:10)	0:03:37 (0:00:13)
	16	0:00:40 (0:00:03)	0:03:00 (0:00:09)	1:20:40 (0:10:27)	6:22:08 (0:12:44)	0:03:36 (0:00:16)
	32	0:00:42 (0:00:05)	0:03:07 (0:00:03)	2:16:51 (0:10:08)	10:53:39 (0:20:22)	0:03:44 (0:00:11)
	64	0:00:49 (0:00:03)	0:03:11 (0:00:14)	4:50:09 (0:20:10)	11:52:16 (0:15:53)	0:04:13 (0:00:19)
	128	0:01:06 (0:00:06)	0:03:05 (0:00:21)	9:00:00 (0:15:15)	16:09:45 (0:14:38)	0:04:41 (0:00:20)
256	4	0:00:35 (0:00:07)	0:03:48 (0:00:04)	0:25:56 (0:00:10)	1:53:39 (0:09:50)	0:02:16 (0:00:16)
	8	0:00:31 (0:00:04)	0:03:57 (0:00:14)	0:52:37 (0:04:14)	4:43:37 (0:10:27)	0:02:14 (0:00:15)
	16	0:00:30 (0:00:03)	0:04:59 (0:00:12)	2:44:45 (0:09:22)	5:37:14 (0:12:18)	0:02:06 (0:00:14)
	32	0:00:41 (0:00:07)	0:04:57 (0:00:17)	4:05:03 (0:10:26)	9:24:15 (0:20:27)	0:02:07 (0:00:06)
	64	0:00:50 (0:00:04)	0:04:41 (0:00:21)	9:28:56 (0:14:12)	12:55:11 (0:19:12)	0:02:31 (0:00:13)
	128	0:01:07 (0:00:04)	0:04:36 (0:00:15)	12:50:44 (0:21:14)	18:17:59 (0:22:47)	0:02:45 (0:00:16)
512	4	0:00:46 (0:00:02)	0:05:20 (0:00:23)	0:34:24 (0:07:17)	2:20:52 (0:09:08)	0:04:16 (0:00:08)
	8	0:00:46 (0:00:12)	0:04:56 (0:00:18)	1:28:01 (0:05:29)	10:55:35 (0:10:32)	0:03:11 (0:00:16)
	16	0:00:53 (0:00:08)	0:06:41 (0:01:13)	6:15:58 (0:11:42)	19:27:41 (0:30:33)	0:02:39 (0:00:19)
	32	0:00:57 (0:00:03)	0:06:58 (0:00:56)	7:50:30 (0:20:18)	/	0:02:38 (0:00:12)
	64	0:01:06 (0:00:10)	0:08:19 (0:02:14)	10:51:19 (0:31:23)	/	0:03:05 (0:00:16)
	128	0:01:10 (0:00:04)	0:10:29 (0:01:03)	13:50:45 (0:24:49)	/	0:03:19 (0:00:16)
1,024	4	0:01:07 (0:00:03)	0:07:34 (0:01:07)	0:32:54 (0:03:22)	3:40:49 (0:11:48)	0:07:27 (0:01:04)
	8	0:01:06 (0:00:10)	0:08:35 (0:00:03)	2:43:18 (0:09:22)	19:37:23 (0:31:06)	0:05:18 (0:00:18)
	16	0:00:56 (0:00:08)	0:09:58 (0:01:11)	6:35:20 (0:13:48)	/	0:04:16 (0:00:14)
	32	0:00:51 (0:00:09)	0:11:23 (0:01:05)	10:15:06 (0:22:22)	/	0:03:54 (0:00:14)
	64	0:01:07 (0:00:04)	0:19:48 (0:03:13)	14:26:26 (0:31:40)	/	0:03:55 (0:00:13)
	128	0:01:33 (0:00:08)	0:21:16 (0:02:14)	/	/	0:04:04 (0:00:16)
2,048	4	0:01:51 (0:00:11)	0:12:48 (0:00:34)	0:36:33 (0:06:20)	11:06:41 (0:19:32)	0:14:32 (0:02:16)
	8	0:01:56 (0:00:08)	0:14:55 (0:02:04)	3:05:18 (0:10:07)	19:24:39 (0:42:49)	0:10:57 (0:00:28)
	16	0:01:37 (0:00:12)	0:15:19 (0:01:06)	10:53:13 (0:13:24)	/	0:07:28 (0:00:15)
	32	0:01:15 (0:00:05)	0:17:42 (0:03:07)	/	/	0:06:46 (0:00:04)
	64	0:01:42 (0:00:06)	0:24:30 (0:02:13)	/	/	0:05:50 (0:00:13)
	128	0:01:56 (0:00:05)	0:27:18 (0:03:09)	/	/	0:06:03 (0:00:15)
4,096	4	0:02:43 (0:00:09)	0:26:37 (0:02:04)	0:46:35 (0:09:46)	20:00:14 (0:19:52)	0:42:32 (0:02:11)
	8	0:02:25 (0:00:03)	0:27:41 (0:02:14)	4:27:07 (0:14:06)	/	0:33:27 (0:04:09)
	16	0:02:24 (0:00:04)	0:36:05 (0:03:09)	16:49:03 (0:28:27)	/	0:22:24 (0:01:15)
	32	0:02:18 (0:00:13)	0:37:51 (0:02:03)	/	/	0:22:00 (0:02:13)
	64	0:02:30 (0:00:08)	0:41:14 (0:02:04)	/	/	0:21:36 (0:01:51)
	128	0:02:44 (0:00:09)	0:52:39 (0:04:09)	/	/	0:10:54 (0:01:03)

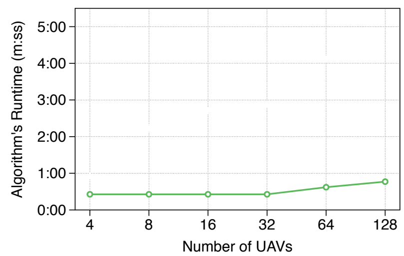


(a) 2 infested palms.
(b) 4 infested palms.
(c) 8 infested palms.
(d) 16 infested palms.
(e) 32 infested palms.
(f) 64 infested palms.
(g) 128 infested palms.
(h) 256 infested palms.
(i) 512 infested palms.
(j) 1,024 infested palms.
(k) 2,048 infested palms.
(l) 4,096 infested palms.

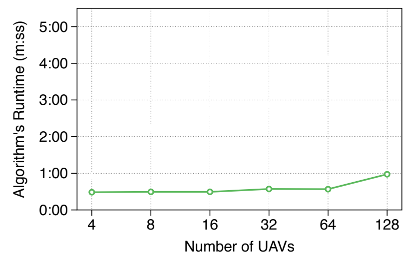
Figure 6: Runtime performance as the number of infested palms increases from 2 to 4,096 in the plantation area.



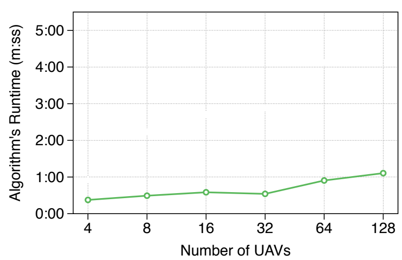
(a) 2 infested palms.



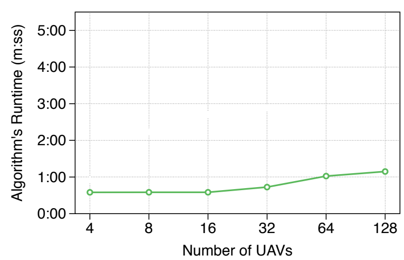
(b) 4 infested palms.



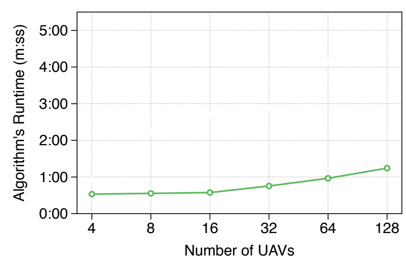
(c) 8 infested palms.



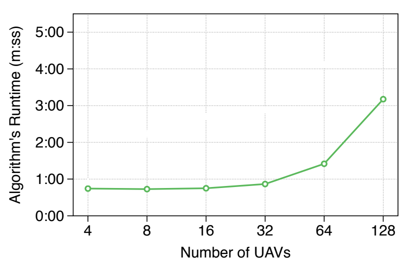
(d) 16 infested palms.



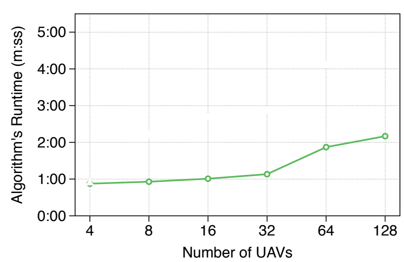
(e) 32 infested palms.



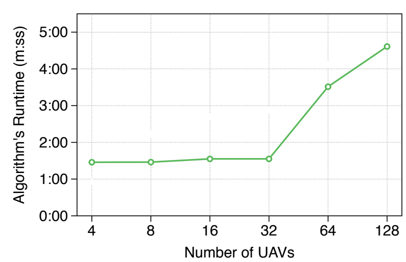
(f) 64 infested palms.



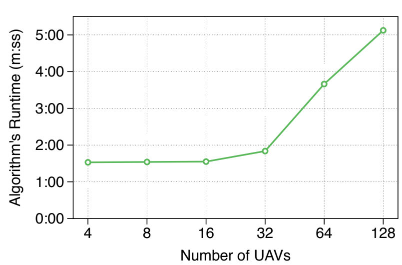
(g) 128 infested palms.



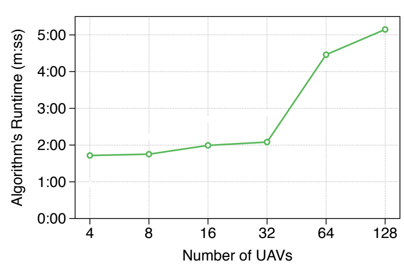
(h) 256 infested palms.



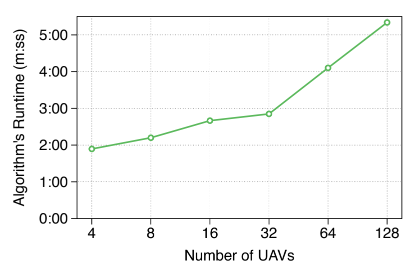
(i) 512 infested palms.



(j) 1,024 infested palms.



(k) 2,048 infested palms.



(l) 4,096 infested palms.

Figure 7: Runtime performance of UTARB as the number of infested palms increases from 2 to 4,096 in a large plantation area.

Table 4: Average runtime performance and standard deviation (h:mm:ss) of UTARB in a small and a large area as the number of infestations increases from 2 to 4,096 infested palms.

Infestations	UAVs	UTARB (small area)	UTARB (large area)
2	4	0:00:15 (0:00:03)	0:00:17 (0:00:08)
	8	0:00:16 (0:00:02)	0:00:18 (0:00:13)
	16	0:00:15 (0:00:04)	0:00:19 (0:00:12)
	32	0:00:16 (0:00:13)	0:00:25 (0:00:09)
	64	0:00:17 (0:00:05)	0:00:34 (0:00:13)
	128	0:00:18 (0:00:08)	0:00:48 (0:00:15)
4	4	0:00:15 (0:00:03)	0:00:25 (0:00:09)
	8	0:00:15 (0:00:04)	0:00:25 (0:00:03)
	16	0:00:19 (0:00:03)	0:00:25 (0:00:04)
	32	0:00:20 (0:00:09)	0:00:25 (0:00:03)
	64	0:00:27 (0:00:07)	0:00:37 (0:00:12)
	128	0:00:31 (0:00:05)	0:00:46 (0:00:15)
8	4	0:00:18 (0:00:04)	0:00:29 (0:00:10)
	8	0:00:19 (0:00:06)	0:00:29 (0:00:09)
	16	0:00:20 (0:00:03)	0:00:29 (0:00:14)
	32	0:00:21 (0:00:02)	0:00:34 (0:00:05)
	64	0:00:22 (0:00:08)	0:00:34 (0:00:14)
	128	0:00:23 (0:00:04)	0:00:58 (0:00:25)
16	4	0:00:19 (0:00:05)	0:00:22 (0:00:03)
	8	0:00:19 (0:00:05)	0:00:29 (0:00:08)
	16	0:00:22 (0:00:02)	0:00:35 (0:00:04)
	32	0:00:29 (0:00:08)	0:00:32 (0:00:12)
	64	0:00:31 (0:00:03)	0:00:54 (0:00:11)
	128	0:00:46 (0:00:06)	0:01:06 (0:00:05)
32	4	0:00:21 (0:00:03)	0:00:34 (0:00:06)
	8	0:00:18 (0:00:03)	0:00:35 (0:00:09)
	16	0:00:25 (0:00:06)	0:00:35 (0:00:07)
	32	0:00:31 (0:00:05)	0:00:43 (0:00:14)
	64	0:00:47 (0:00:04)	0:01:01 (0:00:16)
	128	0:00:48 (0:00:03)	0:01:09 (0:00:15)
64	4	0:00:28 (0:00:02)	0:00:32 (0:00:04)
	8	0:00:31 (0:00:04)	0:00:33 (0:00:04)
	16	0:00:32 (0:00:05)	0:00:34 (0:00:07)
	32	0:00:32 (0:00:06)	0:00:45 (0:00:05)
	64	0:00:30 (0:00:06)	0:00:57 (0:00:08)
	128	0:00:38 (0:00:04)	0:01:14 (0:00:15)
128	4	0:00:40 (0:00:04)	0:00:52 (0:00:02)
	8	0:00:40 (0:00:03)	0:00:55 (0:00:15)
	16	0:00:40 (0:00:03)	0:01:06 (0:00:11)
	32	0:00:42 (0:00:05)	0:01:08 (0:00:12)
	64	0:00:49 (0:00:03)	0:01:52 (0:00:06)
	128	0:01:06 (0:00:06)	0:02:10 (0:00:12)
256	4	0:00:35 (0:00:07)	0:00:52 (0:00:10)
	8	0:00:31 (0:00:04)	0:00:55 (0:00:11)
	16	0:00:30 (0:00:03)	0:01:00 (0:00:08)
	32	0:00:41 (0:00:07)	0:01:08 (0:00:06)
	64	0:00:50 (0:00:04)	0:01:52 (0:00:17)
	128	0:01:07 (0:00:04)	0:02:10 (0:00:08)
512	4	0:00:46 (0:00:02)	0:01:27 (0:00:08)
	8	0:00:46 (0:00:12)	0:01:27 (0:00:12)
	16	0:00:53 (0:00:08)	0:01:33 (0:00:09)
	32	0:00:57 (0:00:03)	0:01:33 (0:00:07)
	64	0:01:06 (0:00:10)	0:03:30 (0:00:18)
	128	0:01:10 (0:00:04)	0:04:36 (0:00:21)
1,024	4	0:01:07 (0:00:03)	0:01:31 (0:00:15)
	8	0:01:06 (0:00:10)	0:01:32 (0:00:16)
	16	0:00:56 (0:00:08)	0:01:33 (0:00:16)
	32	0:00:51 (0:00:09)	0:01:50 (0:00:17)
	64	0:01:07 (0:00:04)	0:03:39 (0:00:12)
	128	0:01:33 (0:00:08)	0:05:07 (0:00:20)
2,048	4	0:01:51 (0:00:11)	0:01:43 (0:00:17)
	8	0:01:56 (0:00:08)	0:01:45 (0:00:09)
	16	0:01:37 (0:00:12)	0:01:59 (0:00:18)
	32	0:01:15 (0:00:05)	0:02:04 (0:00:11)
	64	0:01:42 (0:00:06)	0:04:27 (0:00:21)
	128	0:01:56 (0:00:05)	0:05:08 (0:00:26)
4,096	4	0:02:43 (0:00:09)	0:01:53 (0:00:14)
	8	0:02:25 (0:00:03)	0:02:12 (0:00:16)
	16	0:02:24 (0:00:04)	0:02:39 (0:00:11)
	32	0:02:18 (0:00:13)	0:02:50 (0:00:18)
	64	0:02:30 (0:00:08)	0:04:06 (0:00:23)
	128	0:02:44 (0:00:09)	0:05:20 (0:00:18)