Investigation: An Ant-Inspired Task Allocation Model With a Near-Optimal Distribution of Labor for Swarms of Robots

Thesis by
Alexandre Jean-Pierre Vanini
alva@itu.dk

A thesis submitted in fulfilment of the requirements for the degree of
Master of Science
KISPECI1SE

IT UNIVERSITY OF COPENHAGEN

Main Supervisor:
Payam Zahadat (ITU)
paza@itu.dk

Secondary Supervisor:
Kasper Støy (ITU)
ksty@itu.dk

IT UNIVERSITY OF COPENHAGEN
Copenhagen, Denmark

2021
Defended 11.06.2021

ACKNOWLEDGEMENTS

Let me drop the formalities and jump back into the student I am for a page.

I would like to express my sincere gratitude to my supervisor, Payam Zahadat, for her outstanding support and availability throughout the entirety of this thesis.

I am also extensively grateful to Eline, Joshua, Mathilde, and Sebastien for their time and valuable inputs on the thesis work.

This thesis is not only the result of 4 months of hard and devoted work but of 2 years abroad of my dear hometown, Geneva. I am profoundly grateful to my family and friends who supported me in this time abroad, regardless of the distance.

This thesis marks the end of my study and the beginning of a new chapter.

Thank you. Thank you all.

Alexandre Vanini.

ABSTRACT

Swarm robotics is a vast and fast-growing sector that strives to push the boundaries of robotics. It is believed that it will provide many of the solutions of tomorrow, but with these expectations so naturally follows the challenges of today. Today's literature on the topic goes from robot motion to localization and mapping. The focus of this thesis is turned towards dynamic task allocation in swarms of robots, a dynamic way of distributing a task in relation to the relative demand of the environment. Most of today's research partially solves most of the challenges faced by dynamic task allocation. However, they usually lack versatility as their focus is turned towards a single aspect.

This thesis proposes an investigation of an ant-inspired task allocation mathematical model created by Cornejo et al. [1], which offers a near-optimal distribution of labor for swarms of robots. Heavily inspired by the Response threshold model, the algorithm is based on a binary feedback function that allows a worker to sense its surrounding environment and information about the different needs of the tasks. The model's strength lies in this binary feedback function's abstraction, as one can model it in many ways, enabling extensive research and application opportunities.

The model is successfully implemented in two architectures of the swarm robotics paradigm; A centralized version where a communication tower is used to share the information to the swarm, and a distributed version where the information is instead shared from a robot to another. The model is experimented on in a self-developed simulation environment over a three-dependent task system to reflect on its ability to reach equilibrium in the distribution of labor while maintaining a low energy consumption. Moreover, a random task allocation algorithm, a greedy task allocation algorithm, and another insect-inspired task allocation algorithm are implemented to provide a deeper investigation of the model. The results have shown that both architectures can reach a near-optimal labor distribution regardless of the colony size and the relative workload while keeping a low energy consumption in terms of the number of task switches per robot and the covered distance. Furthermore, analyses of the model through variations of the communication range and noise in the system have shown that it is robust to noise and communication failures.

Although the investigation offers a relatively simple implementation of the binary feedback function, the findings have shown that the model reaches similar performances compared to the other investigated methods. It implies that the model can achieve even better results through incremental improvements.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Number *Page*

LIST OF TABLES

Number *Page*

LIST OF ALGORITHMS

NOMENCLATURE

AITA. Ant-Inspired Task Allocation. The original investigated algorithm.

CAITA. Centralized Ant-Inspired Task Allocation. The centralized version of the original investigated algorithm.

DAITA. Distributed Ant-Inspired Task Allocation. The distributed version of the original investigated algorithm.

GTA. Greedy Task Allocation. A greedy task allocation method.

PSI. Partitioning Social Inhibition. A task allocation algorithm created by Zahadat et al, inspired by honeybees.

RNDTA. Random Task Allocation. A random task allocation method.

TA. Task Allocation.

1. INTRODUCTION

The expectations of robotic solutions are increasing as the technology is being applied in ever novel circumstances, pushing the boundaries of what was thought possible. But in the wake of the technology's increased applications comes the ever more complex challenges and questions about how the technology is best implemented. These questions are especially prevalent in a specific sector growing faster than any other within robot technology, Swarm robotics. Swarm robotics explores many robotic concepts such as communication systems, control approaches, mapping and localization, object transportation and manipulation, motion coordination, robot learning, and task allocation [2]. Task allocation is one of today's biggest challenges as it underlines more significant dilemmas such as optimal task assignments and energy consumption of groups of robots. Moreover:

- Coordination of a large group of individuals is exponentially more challenging as the task's complexity increases and the environment grows.
- Pre-determined task allocation strategies will suffer from a lack of versatility in dynamic environments as they do not have direct tools to respond to spontaneous events.
- Inaccuracy and inconsistency in dynamic systems can exponentially grow if non-resolved, resulting in high energy consumption.
- Dynamic and scalable labor distribution is hard to achieve through limited strategies.

Most of today's researches on task allocation for robot swarms already partially solve these problems. However, they usually lack versatility in their implementation as they are typically designed to work around a single idea. To open new possibilities and design solutions, Cornejo et al. [1] have proposed a mathematical model for dynamic task allocation inspired by ants and their collective social behaviors related to the distribution of labor (AITA). The model provides a framework to study different implementations of task allocation by describing a set of helpers and functions to frame the problem. Their primary mathematical investigation mentioned that their algorithm could maintain a near-optimal labor distribution while keeping a low energy consumption. The purpose and contribution of this thesis are, therefore, to provide one of many possible implementations of the model in a self-developed simulation in order to investigate its performance outside of the mathematical frame and to resolve the challenges mentioned above by:

- Dynamically allocating individuals in relation to the current environmental demand.
- Reducing inconsistency in task switches, thereby reducing energy consumption.
- Being adaptive to a large and undefined number of tasks for large groups of individuals while keeping accurate labor distribution.
- Implementing a solid communication framework robust to communication failure and noise.

Swarm robotics can be defined as « a new approach to the coordination of multi-robot systems consisting of large numbers of relatively simple robots that take its inspiration from social insects. The most remarkable characteristic of swarm robots is the ability to work cooperatively to achieve a common goal » [2]. Moreover, it is "a property of systems of non-intelligent robots exhibiting collectively intelligent behavior" [3]. Swarm robotics highlights significant advantages in scalability, as interactions within the swarm members are local, and thus, robots can leave or join a task at any time without disrupting the process. Moreover, it is highly stable, as if a large part of the swarm quits, the rest of the swarm can still work on the objective. Furthermore, it is economically attractive as the systems are usually composed of cheap robots. Lastly, it is energy-efficient as the small and cheap robot typically consumes less energy in comparison to using a more advanced robot [4]. Notable swarm robotics applications include planetary exploration, victim rescue during natural disasters, agriculture, military, etc.

The studies of task allocation methods related to swarms of robots are mainly used to understand how a complex system of tasks can be solved using multiple agents [5, 6]. The tasks can either be achieved by an individual alone or by a group of robots cooperatively [7]. Distribution of labor can be considered done in two ways: Given that the environment is limited in its evolution, one can implement a pre-determined task allocation strategy, such as robot-planning, where the individuals can perform a set of given tasks but are unable to adapt to dynamic changes. Conversely, one can choose to use dynamic strategies where autonomous robots are given behavior-based mechanisms and adaptive task allocation methods to respond to unplanned events spontaneously. The focus of this thesis is turned towards the latter; Dynamic task allocation strategies.

The investigation of the provided ant-inspired mathematical model is divided into two architectures of the swarm robotics paradigm; Firstly, a centralized architecture, where the information about the environment is not shared among all the individuals but is instead kept in a single entity that any robot can reach out to given deterministic conditions (space and time). This single entity is usually referred to as the leader and can be anything from a robot to a static information center and is also responsible for delivering a task allocation for any robot requesting one. The centralized architecture is well suited for a small number

of robots. However, it has apparent downsides when the number of robots becomes more significant as the communication failure (information loss) and overhead quickly create a disturbance in the system. Moreover, a centralized system has what is commonly referred to as a single point of failure. If the information center breaks or stops functioning, the entire swarm is impacted and cannot perform further action [8]. The main advantages of the centralized architecture are that it is easier, cheaper, and more convenient to implement than other architectures.

The second investigation is in regard to the distributed architecture. This time, the information is shared among all the workers of the swarm through local communication. Each robot shares its state and is responsible for understanding its environment and assigning a task to itself. Given the mode of communication, this architecture does not suffer the same downsides as the centralized one. It is scalable and robust to failure as if one or few robots are removed, the rest of the swarm keeps sharing their state, and the system keeps working. Moreover, this architecture is expected to suffer less from communication failure and overhead as if such happens, it only affects one robot and not the entire communication process [8].

The motivation of using two architectures of the swarm robotics paradigm lies in the fundamental difference that separates them. Both versions have advantages and disadvantages. Therefore, there is a high value in experimenting both to understand how their performances can shape the use of AITA in real-life situations.

The following section *1.1 Related Work* goes through the existing body of work. Then, section *2. Ant-Inspired Task allocation Model* describes the algorithm and the set of assumptions defined by the original authors. The third section *3. Simulation Design* describes the conceptualization and the implementation of the simulation environment and the integration of both the centralized and distributed version of the model. The fourth section *4. Methods* describes the methods and environment used for the experiment. Section *5. Experiments and Results* goes through and reflects upon the result obtained from the experiments. The first set of experiments observes the effects of communication disturbance with respect to different levels of noise and different communication ranges on the system. In additions, the ant-inspired model is compared to 3 other task allocation methods (a random task allocation method, a greedy task allocation method, and another insect-inspired task allocation method) in relation to its task completion rate, its scalability, its optimization of the distribution of labor, its energy consumption and its adaptability to the change of workforce. Finally, the last section *6. Discussion and Conclusion* discusses the results obtain and reflects on possible improvements for future works.

1.1 Related Work

There exists an extensive body of literature and research that has worked on insect-inspired models to solve the dynamic task allocation problem. Jevtić et al. [9] proposed a decentralized algorithm inspired by the Distributed Bees Algorithm (DBA) for target allocation in large groups of autonomous robots where scalability in terms of the demands of the tasks played a role in the task allocation distribution. The experiments showed that the algorithm offers excellent scalability in terms of the number of robots and the number of tasks. However, the model has shown weaknesses in the labor distribution under specific parameter settings where decreasing deployment cost resulted in a higher distribution error.

Zahadat et al. [10] proposed a honeybees-inspired algorithm called the Partitioning Social inhibition Method (PSI) and demonstrated that through local interactions, the method could achieve task equilibrium regardless of the colony size and relative workload. In opposition to the work of Jevtić et al., the PSI algorithm has shown a strong performance in terms of the distribution of the tasks.

Inspired by ants and the Response Threshold Model, Alves et al. [11] proposed methods of self-awareness in robot task selection. In this system, individuals could decide whether to pursue a task or to stop it independently of the task stimuli. The results showed that the swarm could reach equilibrium while maintaining a low energy consumption. However, the model was challenged by how the information flowed throughout the swarm.

As previously mentioned, swarm communication is another extensive area of research in the swarm robotics sector. Ducatelle et al. [12] proposed two methods for decentralized concurrent task allocation for groups of flying robots in a confined area where novel communication methods were used to share the demands of the tasks globally. The first method was based on interaction through light signals-based communication, where robots reacted to different light signals and colors dependent on the current task distribution. The second was a gossip-based method where the infrared sensors present on each robot allowed primitive communication of task status and robot status. The results concluded that gossip-based communication which used infrared sensors was more efficient than using other sensors to detect light emissions from other individuals in a limited environment. However, they also concluded that the gossip-based communication was subject to packet loss and, therefore, less scalable.

Along the same lines, Jamshidpey and Afsharchi [13] proposed a multi-robot cooperative task allocation study in a dynamic, unknown environment through different communication-based approaches. They defined the following four communication-based approaches: The first approach was a static communication-based method where a robot's task allocation remained unchanged until it ran out of search energy. The second approach was an improved version of the first method called the dynamic communication-based approach, in which a robot would choose its next task assignment following a probabilistic

model when running out of search energy. The remaining methods (a decentralized and a centralized one) used Chapar towers (a radio communication station inspiration) placed in the environment, which acted as an information center. In the decentralized version, groups of robots called the "Chapars" transmitted information from the Chapar towers to the workers and so forth. In the centralized version, one Chapar station covered the entire area, and the robots had to report their current status to it. The conclusions drawn were that there is no significantly higher efficiency when comparing the centralized or the decentralized version. However, single points of failure and robustness issues in communication were highlighted.

Wang and Mao [14] proposed a model where the distribution of labor is based on the Optimal Mass Transport theory (OMT). Their algorithm solved dynamic task allocation for large tasks while being low in computation cost and somewhat robust to communication failures.

Communication brings a wide range of problems, from inaccuracy to communication failure. In this regard, Brutshy et al. [6] proposed a communication-less task allocation method based on the interaction rate sensed by the different individuals. Using a set of interdependent tasks, they proved that the method could reach near-optimal task allocation. Yang et al. [15] also proposed a distributed algorithm based on the Response Threshold Model that did not utilize any communication medium. The research showed that their algorithm could successfully adapt to the demand and the tasks growth via a solid labor distribution regardless of explicit communication within the swarm.

Analyzing the existing body of work, there exist clear limitations and challenges in the swarm robotics sector: 1) Optimal distribution of labor for large groups of robots is usually faulty or results in high distribution error or large energy consumption. 2) The communication used within the swarm limits interactions and scalability opportunities. 3) Versatility of the system is rarely seen as made more complex when working with dynamic environments. The following sections hope to propose a solution to the challenges mentioned in the introductory part of this thesis as well as those seen in the existing literature.

2. ANT-INSPIRED TASK ALLOCATION MODEL

Ants are incredibly social insects in many ways. From their adaptability to a great range of environments [16] to the extremely high level of self-organization in colonies, sometimes reaching up to 10 million individuals [17], they are a great source of inspiration for robotic and swarm intelligence. Their most exciting feature is their optimization of labor distribution, from which many scientific articles have taken inspiration to solve today's swarm robotics' challenges [18, 19, 20]. Among these researches, Cornejo et al.'s mathematical framework proposes a near-optimal task allocation based on ant's collective behaviors. At the time this thesis is written (Q2- 2021), it has not been implemented nor tested in a simulation environment. This thesis attempts to give this algorithm a fair software implementation and experimental investigation against other task allocation methods. The algorithm is inspired by the response threshold strategy where the robots react to stimuli in relation to a task. "What is a response threshold? Let s be the intensity of a stimulus associated with a particular task; s can be a number of encounters, a chemical concentration, or any quantitative cue sensed by individuals. A response threshold θ, expressed in units of stimulus intensity, is an internal variable that determines the tendency of an individual to respond to the stimulus s and perform the associated task » [21]. Although, in this implementation, the individuals cannot directly quantify the stimuli.

2.1 Model Definition

To sense variations in the demands of the tasks and react accordingly, Cornejo et al. introduce four quantifier helpers that give a worker the ability to tell whether a task is in energy deficit or surplus:

- $d(T, t)$. The demand for a task t at a given time T. The demand can depend on any aspect of the current environment, such as weather conditions, the colony's location, the current number of ants in the colony, etc.
- $e(T, a, t)$. How much energy[1] a worker a can provide to a task t at a given time T. The energy a worker can provide to a task depends on environmental variables, workers' characteristics, and previous experiences of the specific task.
- $w(T, t)$. The energy supplied to a task, or, the sum of the energy $e(T, a, t)$ currently provided by every worker a performing a task t at a given time T.

[1] As described in Cornejo et al.'s original paper, the energy unit can be any kind of energy (watt, joules, etc.) as long as it is kept consistent throughout the implementation. This thesis does not use any specific unit.

- $q(T,t) = d(T,t) - w(T,t)$[2]. The difference between the current demand for a task and the energy supplied by every worker to the task at a given time T.

2.1.1 The Binary Feedback Function

The binary feedback function $f(T,i)$ is the final element that binds all of the abovementioned functions together. Through this function, a worker can tell whether a task is in energy surplus or energy deficit. Recall the helper function $q(T,t)$, which is the energy difference for a task. The binary feedback function yields *1* if the energy difference for a task is in equilibrium or energy surplus, *-1* otherwise[3] (note that the binary feedback function does not provide enough information for the workers to tell whether a task has reached exact equilibrium). Sensing the energy difference for a task through a binary function means that a worker cannot quantify by how much a task is in energy deficit or surplus. This unavailability of accurate information is critical as it fundamentally shapes the way the task allocation system works. The reason this model includes a function that offers little accuracy lies in the author's intention to reflect how ants react to this kind of stimuli in real life.

2.2 Implementation: Binary Feedback Function

When describing the different helpers of the model in the previous section, there have been mentions of potential "environmental variable" for the demand or "workers' characteristics and experiences" for the energy a worker can supply to a task, which are yet to define. The environmental variables, workers' characteristics, and experience settings are so vast that it is impossible to include them all in the model. Indeed, even Cornejo et al. have decided to leave this choice to someone else who would implement the task allocation model they have designed, as they highlight how the complexity of individual variation quickly results in an intractable task allocation formulation. Intractable problems describe formulations that cannot be solved using efficient algorithms [22]. Intractable problems are commonly referred to as NP-complete problems [23].

The model is investigated with a set of homogeneous robots[4] that share the same skills and characteristics to solve a given task. Sharing the same abilities already narrows down the expectations and reflections around the energy a worker can supply to a task as no robot can perform a given task better than another. Moreover, it is decided that a robot does not have a memory of past experiences, meaning that the energy supplied by a robot to a task can be set to *1* for any robot and any given task. As for the demand, since the simulated environment is controlled and is meant to reflect the use of such a system in places like

[2] In this thesis, $q(T,t) = d(T,t) - w(T,t)$ is referred to as the "energy difference"

[3] The original paper also introduces other binary feedback functions in their further work section, which are not covered in this thesis.

[4] One of the side goals of this thesis was to implement the model in physical agents. This means that the choice of which robot to use for the implementation was limited by the robot ITU owns, a set of Thymio-II.

a depot or a hangar rather than in places challenged by environmental conditions such as a jungle or a desert, its effect is limited. Thereby, one can set the energetic demand of a collectible resource to 1.

The consequence of setting the energy a robot can supply to a task to *1* and the energetic demand of a resource to *1* on the implementation is the following: Suppose a task has a demand of *3* energy. It then needs at least *3* workers to cover the need as each worker can supply *1* energy. Implementations of systems where the energy an individual can supply to a task depends on more characteristics and other variables are further discussed in section *6.1 Future Work*.

2.3 Algorithm Description

The algorithm considers the size of the colony $|A|$ to be a fixed value. All workers maintain a current task *currentTask*, and can be found in one of the five following states: *RestingState*, *FirstReserve*, *SecondReserve*, *TempWorker*, and *CoreWorker*. Moreover, each agent carries a table of potentials Q for each task used to determine which task it will execute next. The table of potentials Q is updated via the binary feedback function. Tasks in energy surplus or equilibrium get a potential of 0, and tasks in energy deficit see their potential increasing, up to 3. Workers first start idle and in the *RestingState*, and as the environment evolves, they fill up a list of candidate task *candidateList*, which contains tasks with a potential of 3. With equal distribution, a worker will choose a task from the *candidateList* and leave the *RestingState* to move to the *TempWorker* state. Moreover, the paper states that:

"Ants in the *TempWorker* state and *CoreWorker* state work on the task specified by *currentTask* (ants in all other states are idle). Specifically, ants in the *TempWorker* state transition to the *FirstReserve* state if there is a surplus of energy in *currentTask*, and otherwise transition to the *CoreWorker* state. Ants in the *CoreWorker* state transition to the *TempWorker* state if there is a surplus of energy in *currentTask*, and otherwise remain in the *CoreWorker* state. The result is that when there is a surplus of energy all ants in the *TempWorker* state will become idle before any ants in the *CoreWorker* state. Ants in the *FirstReserve* state and *SecondReserve* are state idle, but unlike ants in the *RestingState* (which are also idle) if they start working they will do so at the task they were last working on. Ants in the *FirstReserve* state transition to the *RestingState* if there is a surplus of energy in *currentTask*, and otherwise they transition to the *TempWorker* state with constant probability or join the *SecondReserve* state. Ants in the *SecondReserve* state transition to the *RestingState* if there is a surplus of energy *currentTask*, and otherwise transition to the *TempWorker* state" [1]. For more detail, refer to the original paper.

The complete original algorithm is shown below in *Algorithm 1: AITA Algorithm*.

Algorithm 1: AITA Algorithm

Initialize: $state \leftarrow Resting, currentTask \leftarrow \perp, Q[\tau] = 0, \forall_\tau \in T$
case *state* = *Resting* **do**
    $$\forall_\tau \in T, Q[\tau] \leftarrow \begin{cases} 0, & \text{if } \mathtt{f}(\tau, i) < 0 \\ max(Q[\tau] + 1, 3) & \text{if } \mathtt{f}(\tau, i) > 0 \end{cases}$$
    candidateList $\leftarrow \{\tau \in T | Q[\tau] = 3\}$
    if *candidateList* $\neq \varnothing$ **then**
        if `randInt(0,1)` == *1* **then**
            $\forall_\tau \in T, Q[\tau] \leftarrow 0$
            currentTask $\leftarrow$ random task from candidateList
            state $\leftarrow$ TempWorker
        end
    end
case *state* = *FirstReserve* **do**
    if `f(`*currentTask,i*`)` < *0* **then**
        state $\leftarrow$ Resting
    else
        if `randInt(0,1)` == *1* **then**
            state $\leftarrow$ TempWorker
        else
            state $\leftarrow$ SecondReserve
        end
    end
case *state* = *SecondReserve* **do**
    if `f(`*currentTask,i*`)` < *0* **then**
        state $\leftarrow$ Resting
    else
        state $\leftarrow$ TempWorker
    end
case *state* = *TempWorker* **do**
    if `f(`*currentTask,i*`)` < *0* **then**
        state $\leftarrow$ FirstReserve
    else
        state $\leftarrow$ CoreWorker
    end
case *state* = *CoreWorker* **do**
    if `f(`*currentTask,i*`)` < *0* **then**
        state $\leftarrow$ TempWorker
    end

2.4 Algorithm Correction

Throughout the implementation and tests conducted on the original algorithm, it has been found that there was an error with the increase of potential for a task in table Q, in lines 3 and 4 - *Algorithm 1: AITA Algorithm*. Indeed, imagine the following quite likely situation: All tasks are in energy deficit, and one (or more) worker is in the state *RestingState*.

When a task is in energy deficit, the algorithm will increase its potential in the table of potential Q. However, the algorithm yields that the new value is either *3* or higher, which is problematic because a task can only enter the *candidateList* given that its energy potential is of *3*. Furthermore, it is not given that the task is selected. Indeed, workers select a new task with a probability of ½, meaning that a task with a potential of *3* can re-enter the process and see its potential increase to a value higher than *3*. A value higher than 3 for a task's potential defeats Cornejo et al.'s definition of the model, which states that "The potential for every task is a two-bit value 0, 1, 2, 3". Furthermore, if all tasks in the table of potential end up with a value higher than 3, they cannot be added to the *candidateList*, leading the robots to be stuck without any assignment until a task eventually ends up in energy surplus, resetting its value to 0.

To overcome this situation, line 3 from *Algorithm 1: AITA Algorithm* has been corrected to:

$$\forall_\tau \in T, Q[\tau] \leftarrow \begin{cases} 0, & \text{if } \mathtt{f}(\tau, i) < 0 \\ Q[\tau] + 1 & \text{if } \mathtt{f}(\tau, i) > 0 \end{cases}$$

And line 4 has been corrected to:

$$\text{candidateList} \leftarrow \{\tau \in T | Q[\tau] \geq 3\}$$

This way, the task assignment can be satisfied. The algorithm works as intended by the original creator, and a worker cannot be stuck without a task assignment when tasks are in energy deficit.

2.5 Model Assumptions

Cornejo et al. define a satisfying task assignment as one in which no task is in energy deficit. In other words, they seek a task assignment where the set of task reach equilibrium – I.e., a task assignment where the energy supplied to each task equals the demand of each task. To evaluate this performance, they define an **optimal task assignment** as one that **minimizes** the squared difference between the energy demand of a task and the energy supplied to the task.

3. SIMULATION DESIGN

In order to investigate the ant-inspired task allocation model, a software simulation that reproduces a physical environment has been built[1]. If the choice has been turned to a self-developed simulation rather than using an existing simulator such as Argos[2] or NetLog[3], it is for the following reasons; Firstly, Argos and other simulation environments are heavy, and the author's computer had trouble running them. It would not have been a problem for one or two simulation runs, but the experiments and the drawn conclusions are the results of some 300 simulation runs. Secondly, reproducing an environment is the key to total control over its characteristics and behaviors. Thirdly, the challenges offered by building the simulation are a great way to demonstrate the skill set obtained by the author during his two years of master in computer science with a robotic-oriented specialization. This section is a technical deep-through of the critical points of the simulator's creation, from its conceptualization to its realization (note that only the key points are described, not the full implementation).

Simulation Need. In order to create a simulation environment, the simulator must be able to reproduce a world bound in all its directions, physical agents, physical agent's motions and features, and reproduce key components for robot sensing such as areas of interest and resources to collect.

3.1 Agents

The simulated agents are a set of Thymio-II [24] (a pre-built ready-to-use robot equipped with built-in sensors and an IR communication system). Originally, a Thymio-II robot is equipped with many built-in features such as a temperature sensor, a loudspeaker, an accelerometer, and many others. As the simulation is self-developed, the author chooses what is useful and what is not. The following list describes the implemented features:

- A body.
- Two wheels
- 3 out of 5 front proximity sensors.[4]

[1] `github.com/alevani/master_project`
[2] `https://www.argos-sim.info/`
[3] `https://ccl.northwestern.edu/netlogo/`
[4] the left-most, center and right-most proximity sensors are enough for wall and robot avoidance

- 2 rear proximity sensors.
- 2 ground sensors.

This feature extrapolation alone is enough to reproduce the behaviors needed by the experiments. Figure 3.1 depicts the flow of an agent throughout a simulation run. The CAITA (in green) and DAITA (in red) communication blocks are depicted in the section *3.6 Model Integration*. The blocks (in purple, yellow, and blue) describing the different task flows are shown in Appendix A.

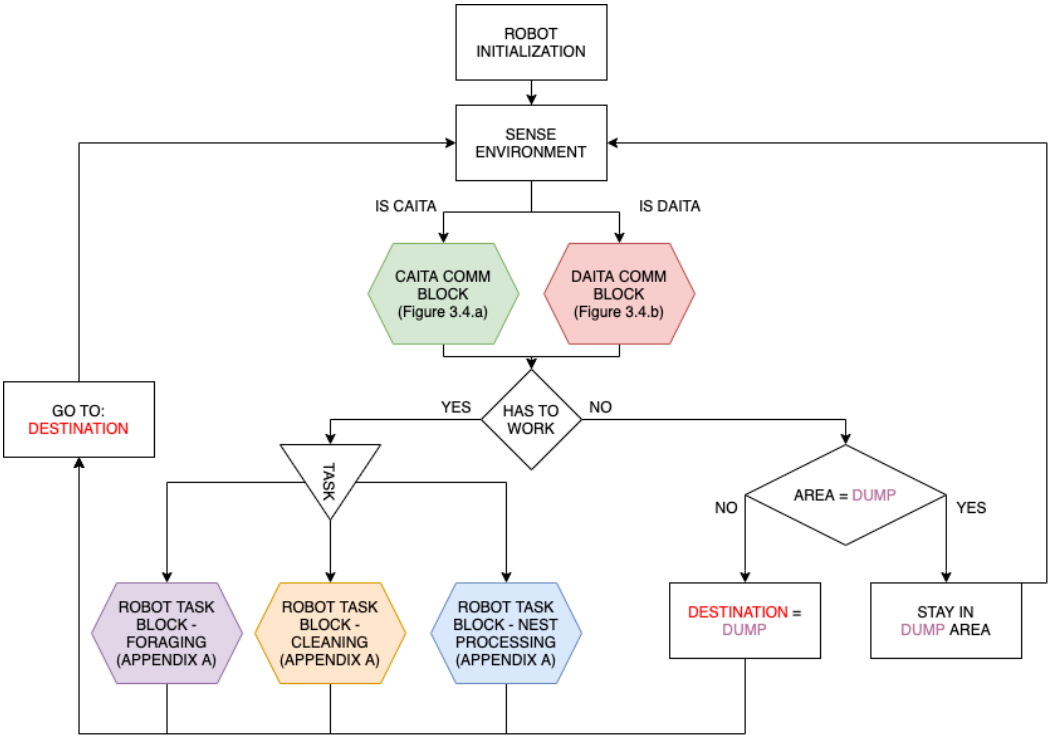


Figure 3.1: Flow of an agent throughout a simulation run

The group of agents evolves in a synchronous environment. The main advantage of building a system based upon synchronous operations is the following: As one does not have to spend time struggling to work with multi-thread interaction and issues related to concurrent programming, implementing the system is faster - leaving more time to program the model and the experiments. Although a synchronous simulator has advantages, it can also be seen as being more deterministic. To counter this and to ensure more accurate reproduction of real events, mechanisms have been implemented:

- The list of robots is shuffled at each new round so that the order of execution is different each time.
- Communication between robots is randomized. In order to simulate this behavior, a random transmission receiver mechanism is implemented and randomly decides from

which incoming transmission a robot will receive its packet[5]. This system ensures fairness and close-to-reality communication transmission.

3.1.1 Motion

In order for the robot to move around in the environment, motion via wheel propulsion is implemented. To recreate best the movement of a real Thymio-II, kinematics equations[6] are used to determine its future position given its current position, its current heading, and its current velocity.

3.1.2 Collision

The simulation uses a Python module called Shapely[7]. The shapely module is used to create geometrics in the simulation, which enables collision detection. To do so, each agent is given a collision box that can collide with the different walls of the arena and the different robots. In addition, the ground sensors that the robots are equipped with are abstracted into two collision boxes, which can then interact with areas and points of interest.

3.2 World

The simulation (see figure 4.2) can reproduce a plain world of unlimited width and height, though each edge is considered a wall. As the agent evolves in a coordinate plan where x and y are in meters, expressed in floats, a world grid represented by a 2-dimensional array is created. This 2D grid is the extension of the visible area, where each point in the agent's world coordinate system is mapped to the array coordinate system with a 100x factor[8]. This 2D array is advantageous as it creates an interactive interface between the agent and whatever is placed in the 2D grid. Before this 2D grid was designed, each sensor would skim through a list of points of interest and calculate whether their absolute position would enter in contact with it, in which case it would signify that a sensor was on top of a resource. If the list implementation has been abandoned, it is because lookup times in lists take up to $O(n)$ (where n is the size of the list), whereas the lookup time of an array is $O(1)$. As the implementation became more extensive and more objects were inserted into the list, the running time became extensively longer.

[5] The distribution is uniform; each packet is given the same chance to be received.

[6] The kinematics equations are also dependent on the wheel dimensions, and the distance that separates them.

[7] `https://pypi.org/project/Shapely/`

[8] If a robot's coordinate is (x =1.45m, y =13.12m), its position within the global 2D array is (x =145, y =1312)

3.2.1 Areas

Areas can be added to the world (see figure 3.2) and can be recognized by an agent via its ground sensors positioned at the floor of its body. Moreover, an agent can detect and remain within a specific area.

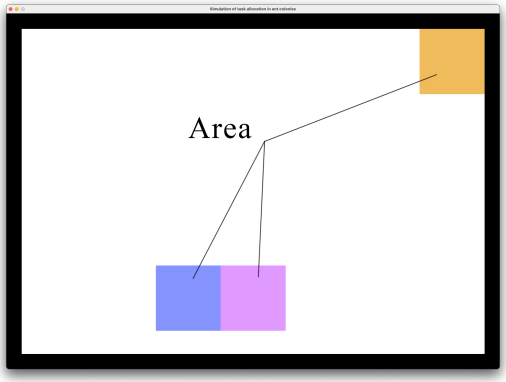


Figure 3.2: Simulation environment, multiple areas

Multiple areas can be seen, each of them represented by a specific color. As the simulated agent is equipped with ground sensors, it can recognize a specific area by its color.

3.2.2 Point of Interest

A point of interest is an object positioned in the world grid. The object, of size 1x1 pixel, can be related to anything a robot could interact with. Moreover, this object carries information such as its state, its position in the world, and its type. By checking its current world grid position, a worker can tell whether one of its sensors is nearby or a on point of interest using simple algorithmics. A point of interest can be picked up by an agent and dropped anywhere in the world grid.

3.3 Navigation

Navigation is one of the most essential features of the simulation. One could think random movements are enough, but in fact, whenever a robot encounters a point of interest, it needs to navigate back home, that is, to a specific location. In order to recreate a good navigation system, many designs have been experimented. The first iterations of the simulation contained a path-following navigation system where agents could sense "pheromones" (placed in the world grid in the form of points of interest) and follow them (see figures 3.3 and 3.4). Although this solution worked well for a robot to follow back its route when encountering a resource, it only did so when there was only one of them. Indeed, as the simulation needs to run many robots, pheromones-based navigation becomes inconsistent, inaccurate, and hard to realize. This system works with real ants because ants can crawl on top of each other.

In the robotic world, such behavior is hard to imagine. Thus, if a robot would encounter another robot in its journey to follow the pheromone path, it would avoid it, lose track of the path, and get lost. Mechanisms for robots to find their lost path back and keep following it are possible, but considering the number of robots, the time allocated for this thesis, and the complexity of the task, it has been decided to abandon the implementation.

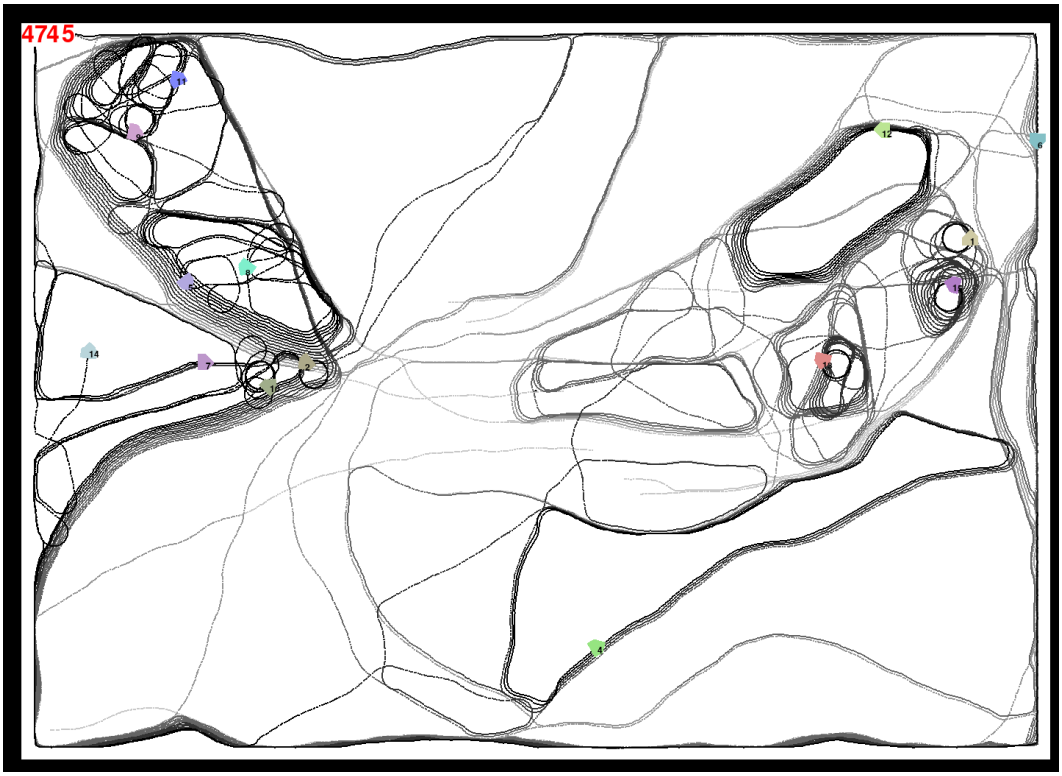


Figure 3.3: Simulation environment, navigation with pheromones

The figure shows multiple robots wandering randomly until they eventually find a path to follow. Multiple behaviors can then be seen, such as path formation or circles of death.

Instead, a global world coordinate navigation is implemented. This time, the robot can move to a given (x, y) coordinate. This solution is more life-feasible as it is possible to implement this mechanism with solutions such as a particle-localization-based algorithm that provides the robot with its somewhat precise position on the plan [25] given that it is equipped with a Lidar or other kinds of feature mapping sensors. The simulation reflects the implementation of such a system but does not implement any advanced localization system. Instead, it assumes that the robots can accurately know their position in the world and move to a given (x, y) coordinate.

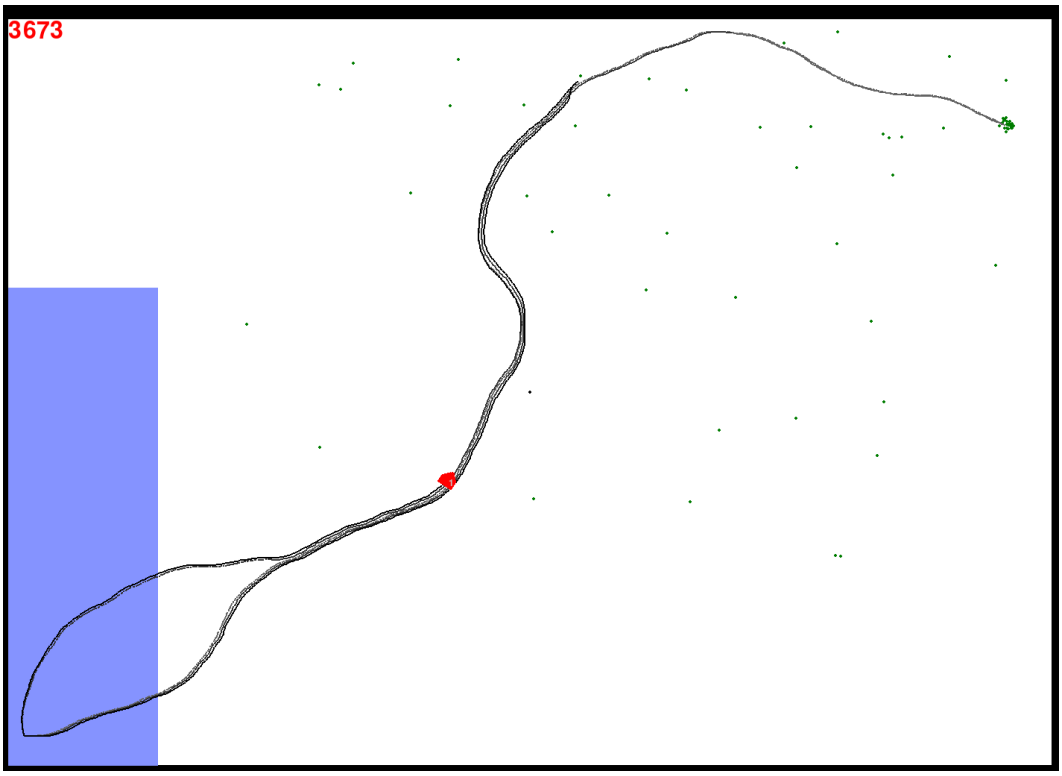
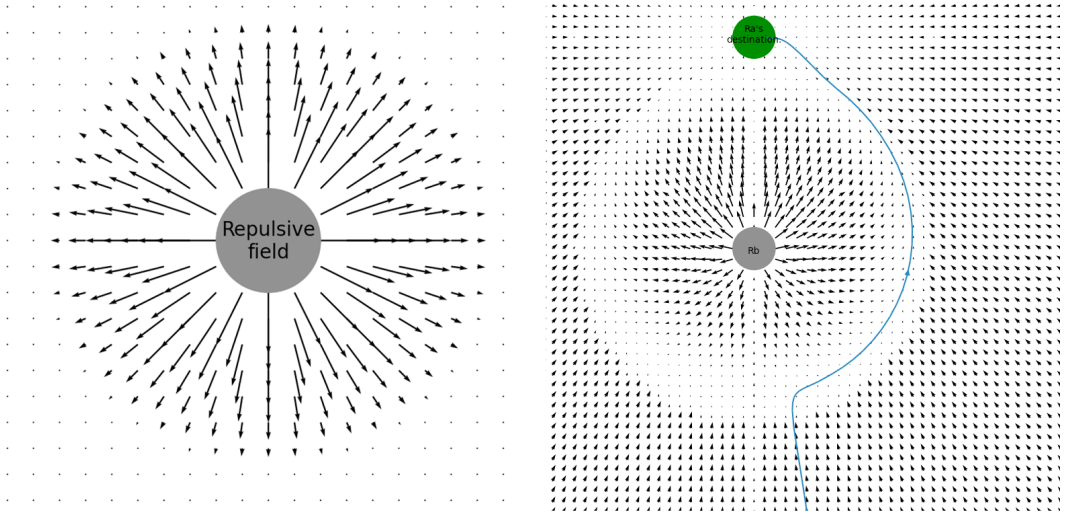


Figure 3.4: Simulation environment, navigation with pheromones to pick up a resource and bring it back home

A robot (in red) wanders outside to gather resources. Once it finds a resource, it goes back home. Once it reaches home, it drops the resource and goes back to the resource's location following its pheromone path, and so on.

3.4 Avoidance System

The avoidance system implemented is based on the proximity sensors placed at the front of a Thymio-II. The designed technic is inspired the by potential fields used in the reactive paradigm [26], which are motor behaviors to avoid objects in a world. In this simulation, each robot acts as a repulsive potential field (see figure 3.5.a). Each time a robot A, R_A, gets too close to another robot, R_B, it is repulsed by its field in relation to its distance from the center; that is, the closer R_A gets to R_B, the greater the repulsion is. Moreover, the position of R_B in the surroundings of R_A determines where R_A goes to avoid collision (see figure 3.5.b). For instance, if R_B gets too close to the upper-left corner of R_A, then R_A will be avoiding R_B by steering to the right. The avoidance system is also used in the navigation system, as when a robot goes to a specific location, it needs to be able to avoid robots it might encounter.



a) b)

Figure 3.5: Repulsive field, visualization

a) Classic repulsive field. b) Depicts the path that a robot takes to reach its destination (in green) if it encounters another robots (in grey) on its way.

3.5 Visualisation

Written with PyGame[9], the visualization (figure 4.2) is built on top of the simulation. It helps visualize what is currently happening and gives precious information such as the path covered by each robot and their position in the plan. Moreover, it is a critical element of the development as it enables the programmer to see problems with its own eyes, which would be more complex only by looking at robot's coordinates printed in a terminal.

3.6 Model Integration

This section describes how the AITA algorithm is integrated into the simulated agents and what design choices have been made. The algorithm discussed in section *2.3 Algorithm Description* is only a brick of the global implementation. Therefore it is essential to explain how the functions of the individuals are designed and how they shape the algorithm's performances. Among others, the feedback function and how it is integrated are discussed. Moreover, this section covers the design of the communication system for the centralized and the distributed version of AITA.

[9] `https://www.pygame.org/news`

3.6.1 Integration: Binary Feedback Function

The binary feedback function proposed by Cornejo et al. is at the core of AITA. It allows workers to access information about their immediate environment on what task is in energy deficit or surplus, which AITA relies on to generate a new task assignment. Therefore, a near-perfect implementation of the binary feedback function is crucial to obtain the best results. The *1.1 Related Work* section is an excellent source of inspiration to design the binary feedback function.

The goal of the integration is to find a way for a worker to obtain crucial information on the current status of each task[10]. As the world is limited in its global information, direct access to data on task needs is hidden from the swarm. Therefore, the sub-goal is to create a communication medium that enables the worker to share their current knowledge of the world and their current progress to other robots. A first method proposed by Ducatelle et al. [12] would be to use light signal-based communication where light emitters and sensors would be placed on top of each agent. The light would then be emitted in relation to the stimulus sensed by each robot in relation to their current knowledge of the task and received by other robots for interpretation. However, methods relying on directional sensors and explicit signal discovery are less accurate and prone to noise and faulty behavior [9].

Another method would have been to use a robot-interaction-based communication method such as in [6], where robots would determine the world situation as a function of the number of encountered robots. Although this method has shown remarkable results, it is limited in the size of the information it can share throughout the swarm and its versatility, adding unnecessary complexity to the implementation.

The method used for this thesis is radio-telecommunication through transmission towers such as in Jamshidpey and Afsharch's work [10], where radio towers are placed on top of each agent to cover a wider communication range than basic built-in IR-sensors. This implementation works well and allows the integration of both a centralized version of AITA, where each agent communicates with a central radio transmission tower given that it is in the coverage area, and a distributed version, where each robot can be equipped with such radio transmission towers and communicate with one another. This thesis does not attempt to cover an in-depth proof of the real-life feasibility of such a communication system. However, it instead uses it as a framework for communication that can be improved or changed.

[10]The binary feedback function also includes the energy a worker can supply to a task. However, as discussed in the section *2.2 Implementation: Binary Feedback Function*, a robot can only supply an energy of 1 to a task. Therefore, no specific integration is needed.

3.6.2 Integration: Centralized Ant-Inspired Task allocation

The centralized ant-inspired task allocation implementation, or CAITA, works as shown in figure 3.6.a; At initialization, starting values for demands are assigned to the information center. As the simulation goes, robots have to periodically drive back to the information center area and report their current status. The status is a piece of knowledge a robot detains. It consists of its identification number, its current task, whether it is currently working, and how many resources it has processed in each task since its last report. The robots are only aware of what is happening in their bubble and have no knowledge of the world and its current situation. Once the information center receives the packets containing the robot's status, it updates its table of global knowledge. The table contains the most recent status report from all robots. It then decides on and sends back a new task assignment to the sender[11]. **Note that:** As the communication between a robot and the information center is two-way, it might happen that a robot successfully sends its report (thereby updating the table of global knowledge), but fails to receive its new task assignment (either due to noise in the communication or other kinds of failure). In CAITA, the transmission tower/information center uses the binary feedback function to determine (along with its table of global knowledge) whether a task is in energy deficit or energy surplus.

Mechanisms for improving the robustness of the system against potential robot failures are implemented. Each time a robot does not report within a specific number of simulation steps, it is considered gone. Its status in the global table of knowledge switches to "gone", which changes the energy supplied to the task the missing robot was performing, leaving a place for another potential *idle*, *firstReserve*, or *secondReserve* robot to take over its work.

3.6.3 Integration: Distributed Ant-Inspired Task allocation

Significant changes are made in the distributed version of AITA, DAITA (shown in figure 3.6.b). This time, the memory is not contained in a single place but shared throughout the entire swarm. At initialization, each robot is given the starting demand for each task, stored in the *demandArray*. Along with it, each robot holds a memory of knowledge about the other robots, called the *memoryArray*. At each simulation step, each robot broadcasts its current status to all other robots. The current status contains the robot's identification number, its current task, its state, and its current progress on each task. Once received by another robot, these pieces of information are processed and stored in the *memoryArray* of the receiver. As the simulation goes on, the information of one robot is broadcast to all other robots, and the knowledge of the world is kept accurate and up-to-date. Before reaching the end of its simulation cycle, a robot runs the AITA algorithm with its current knowledge of the world and is attributed a new task (or not) depending on it. In DAITA, the binary

[11] Note that: a robot is not assigned a new task if it carries a payload

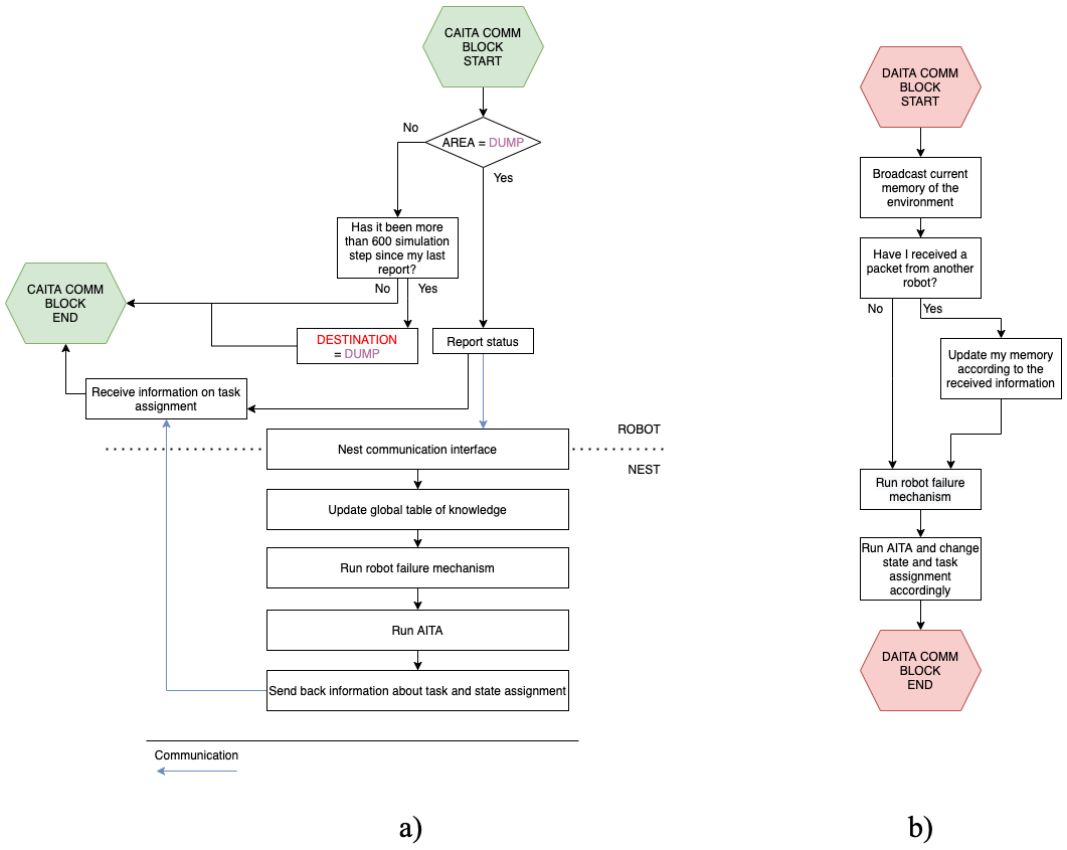


Figure 3.6: Model integration: Communication flow of CAITA and DAITA

a) CAITA's flow diagram. The diagram is separated into two parts: A top part representing the robot's brain, where in-robot logic goes, and a bottom part representing the information tower's flow where logic on information and task assignment goes. b) DAITA's flow diagram representing the in-robot logic of the information + task assignment logic.

feedback function is implemented in each robot, which uses it given its *memoryArray* to determine whether a task is in energy deficit or energy surplus.

Robustness mechanisms similar to the one implemented for CAITA are also present in DAITA. This time, each robot keeps track of the last time another robot has contacted it. Suppose this elapsed time goes over a specific pre-determined simulation time. In that case, the robot is considered gone from the other robot's system. The energy applied to the task the missing robot was performing decreases, leaving room for another potential *idle*, *firstReserve*, or *secondReserve* robot to take over its work.

4. METHODS

This section intends to provide the reader with a description of the method used to investigate the performances of AITA. Moreover, it gives an in-depth specification of the environment within which the algorithm has been experimented (tasks the systems have to perform, areas, and collectibles).

4.1 Introduction

The experiments intend to investigate the efficiency of AITA over five categories that are commonly referred to as being what a swarm robotics system should be good at, namely:

- Its *scalability*, which is the system's ability to adapt to a change of workforce (whether it is adding or removing individuals).
- Its *robustness*, or how well the system does against communication or robot failures.
- Its *versatility*, which states that the system should apply to a wide range of tasks.
- Its *adaptability*, or how good can the system adapts to dynamic environments.
- Its *reliability*, which expresses that a system should be consistent in its probability of solving a given task.

In order to investigate the system upon the abovementioned categories, two investigation methods are used: Investigation by comparisons and investigation by metric evaluation.

4.2 Investigation by Comparisons

There exist many methods of algorithm evaluation. One is to search for and implement an algorithm designed to solve the same challenges as AITA and is relevant for comparisons. This section provides an overview of RNDTA, GTA, and PSI; three algorithms chosen to be compared against AITA. These algorithms are implemented in the same simulation environment as AITA and go through the same experiments.

4.2.1 Random Task Allocation Algorithm

The random task allocation algorithm, or RNDTA, is a method where each individual is randomly attributed a new task every 600 simulation steps[1] following a uniform distribution. Since the robots are not required to share any information and do not need to be aware of the current world state, this task allocation system does not suffer any communication failure or overhead, making it highly scalable and robust.

This algorithm is chosen as it is the simplest one can design. Thus, it serves as a lower boundary for what AITA (and any other elaborated algorithm) should not go below in terms of performances. It is expected that this algorithm performs the worse as it does not take into account the current state of the world and is very inconsistent.

4.2.2 Greedy Task Allocation Algorithm

The greedy task allocation algorithm, or GTA, is a system where the robots share their states to others within the swarm and coordinate to cover the task that requires the most attention. The memory and communication systems are the ones used by DAITA. Using the same communication and memory system as DAITA means that the information is distributed among the entire system. The workers are then prone to suffer the same challenges as DAITA; communication failure and system disturbance.

A greedy algorithm has been selected because it performs well in a wide range of situations but is expected to have high consumption in terms of distance covered and task switches per robot.

4.2.3 Partitioning Social Inhibition Task Allocation Algorithm

The Partitioning Social Inhibition task allocation algorithm, or PSI, is a system issued from the research paper "Division of Labor in a Swarm of Autonomous Underwater Robots by Improved Partitioning Social Inhibition" (Zahadat et al. [13]). Zahadat et al. claim that "The PSI algorithm maintains a division of labor and allocation of tasks to different members of a swarm. It is adaptive to changes in the swarm size and relative demands for different tasks" [13]. Its adaptivity in terms of the swarm size and the relative demand is an essential aspect of modern task allocation methods. Moreover, as PSI has explicitly been written to solve the task allocation problem (unlike GTA or RNDTA), it opens a new door to reflections and improvements in relation to AITA's performances. The following section gives an overview of how the algorithm works. In addition, a description of its implementation in the environment conceived for this thesis elaborates on how PSI's specific variables are adapted to the system.

[1] After thorough experiments, 600 simulation step has been seen to be the most optimal task time

Author's note: For this thesis work, Payam Zahadat has been kind enough to provide me with the C++ code with which their experiments were conducted. The algorithm has then been transferred to the Python code I am working with and adapted to the current environment and communication mechanisms. This adaptation means that PSI executes the same set of tasks as all other TAs, which provides fair and accurate data. Overall, the algorithm is expected to perform as well as in Zahadat et al.'s experiment. However, the system is also expected to suffer from applying the algorithm to my environment as it was first designed to run in a completely different setup.

Algorithm Description

Each robot of the swarm holds an x value that represents their physiological age - the real biological state of a living thing. In the original paper, the analogy is made between the robots and honeybees, from which PSI is inspired. This x value is distributed over a range from x_{min} to x_{max}, where the range is split equally by the number of tasks so that each task gets the same amount of distribution (as shown in figure 4.1). PSI aims to distribute each individual's x value relative to the current demands for the tasks to achieve equilibrium (recall that equilibrium is when the number of robots assigned to a task matches or covers the current demand of the task). PSI uses the same communication system as DAITA and is thus distributed. Using the same communication system means that PSI is expected to suffer from the same challenges as DAITA (communication failure and system disturbance). The value x changes through time and local interactions with the swarm members. Please refer to the original paper for further information on how local interactions within individuals changes the x values.

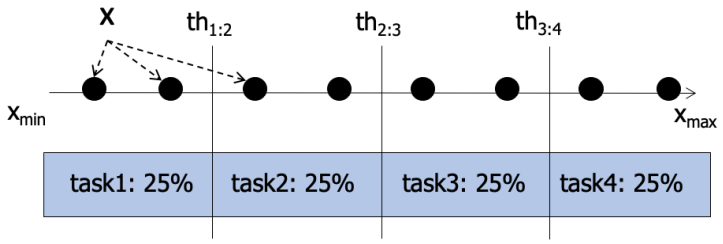


Figure 4.1: PSI's distribution of the task's thresholds

Algorithm Integration

As mentioned above, PSI has not been designed to run under the same environment as AITA's. For instance, in their original paper Zahadat et al. assume that the demand is constant or only variates slightly but independently of environmental conditions or task completion. These variations mean that PSI's relative workflow has been adapted to fit the

constraints imposed by this thesis' framework. The two impacted areas are the demand and the specific condition under which a worker can be allocated a new task.

Demand. This thesis' implementation of the environment yields that the demand for a task can vary from $-\infty$ to ∞. In contrast, in Zahadat et al.'s system, the tasks have a value representing a fraction of the current demand variating between 1 and some positive number. The independent variation of the demand means that PSI has not been written to handle demands of 0 or negative demand. Thereby, the integration implements a function that maps the demand system from AITA to a 1 - 20 PSI scale.

With this mapping, if the sequence of tasks' demands in AITA is *[23, 132, 12]*, it is then mapped to *[4, 20, 2]*. **Note that:** Negative numbers are mapped to 1.

New Task Assignment. As for all the other algorithms, a robot using PSI cannot be given a new task if it is currently performing one. It means for PSI that the changes in the x value of each individual are delayed as long as the worker is currently carrying a payload.

Algorithm Initialization

PSI describes a set of initial variables that one can act on which changes the algorithm's performances and behavior over time. The PSI variables used for the thesis implementation are described in table 4.1. For more information on what each variable means and its influence on the PSI system, refer to Zahadat et al.'s original paper.

Name	Value
x_{min}	0
x_{max}	512
noise on x	0
δ	12
ϕ_{base}	0.3
l_u,l_b	3
thresholds	equal segments
initial x	2
initial d_{lower}	$x - x_{min} - 1$
initial d_{higher}	$x_{max} - x - 1$

Table 4.1: PSI's variables initialization

Author's note:. A wide range of tests and experiments conducted on the adaptation of PSI over AITA's environment have proven that the implemented system accurately complies with how the PSI's task allocation algorithm is designed to work. The tests and experiments are not shown in this project, as the goal is only to achieve similar performances to PSI under its original environmental setting.

4.3 Investigation by Metric Evaluation

The second method used is an investigation by metric evaluation. Different metrics are considered: the task completion rate, the total covered distance by all robots, the distribution of robots over the different tasks, the number of task switches per robot, and the demand/energy supplied for each task. Moreover, each system keeps track of the average sensed demand by the swarm – or the swarm's perception of the current demand for each task –, and the real demand, i.e., the real value of the demands. The two metrics are then turned into a new metric; The "swarm's perception error E":

$$E(\tau, t) = \sum_{t \in \tau} |d(T, t) - D(T, t)|,$$

where $d(T, t)$ is the sensed demand by all robots (averaged) for a task T at a given time t, and $D(T, t)$ is the real demand for a task T a the given time t

The swarm's perception error E also describes the speed at which the information of the demands of the different tasks is shared throughout the swarm. *Note that:* When using the metric E, the term used for CAITA will be "the nest's perception error" as in this system, only the information center (the nest) knows about the current state of the world.

In addition, the model is tested upon Cornejo et al.'s metric of stable task allocation system defined in section *2.5 Model Assumptions* – the minimalization of the squared difference between the energy demands of a task and the energy supplied to the task. **Note that:** Throughout this thesis, the "minimalization of the squared difference" might be referred to as the "labor distribution optimization" or "labor distribution optimization metric".

4.4 Investigation Environment

The different task allocation methods are tested and experimented on in an agent-based simulation. The model of the simulation (figure 4.2) consists of a 2D environment wide of 10 meters and tall of 7 meters, populated with four types of agents (idle, foragers, nest processors, and cleaners), a nest including three main areas (or chambers), the dump area (shown in blue, 1.4 meters x 1.4 meters), where resources collected from the outside world are stored (the area is also used as the information center described in section *3.6.2 Integration: Centralized Ant-Inspired Task allocation*, where robots report their status). The transit area (shown in pink, 1.4 meters x 1.4 meters), where resources processed from the dump area are stored. Finally, the waste area (shown in orange, 1.4 meters x 1.4 meters) where resources stored in the transit area are trashed. Everything that is not one of these three areas is considered a foraging area where resource items are distributed at the start of

each simulation following a random uniform distribution[2]. The topology of the world is a rectangle box bounded in all its directions which no agent can traverse.

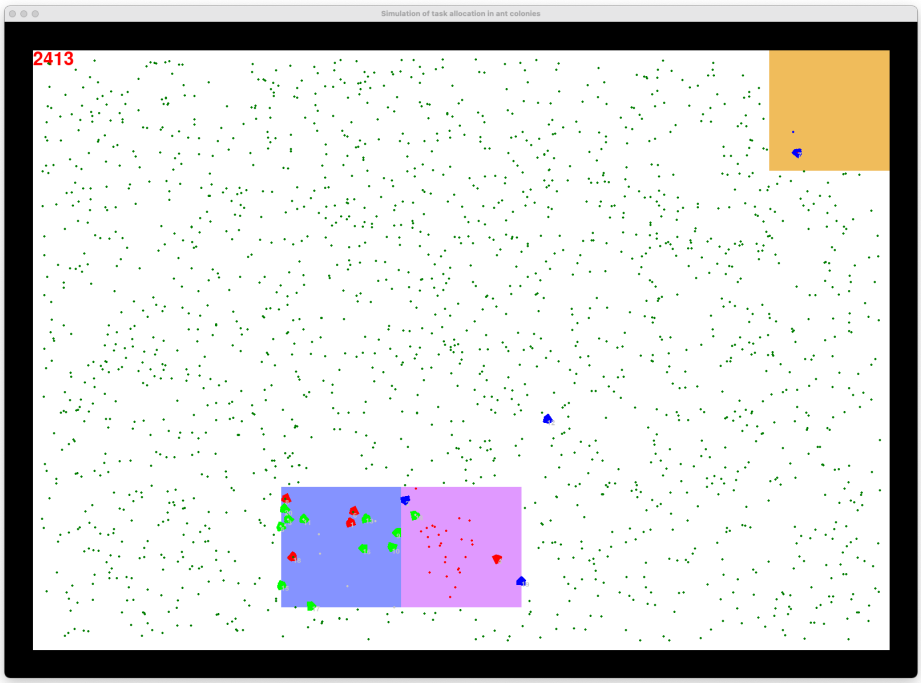


Figure 4.2: A snapshot of the simulation environment

Workers can carry out four kinds of tasks depending on the demand of the colony: Firstly, the idle task – which consists of being at rest in the nest (not moving) waiting to be attributed a task. Workers carrying out the idle task are shown in black. Secondly, resource collecting or foraging - where workers wander outside the nest to collect resources that they bring back to the dump area. Workers carrying out the foraging task are shown in red. Thirdly, nest processors – which consists of processing the resources brought back by the foragers and move them to the transit area. Workers carrying out the nest processing task are shown in green. Finally, the cleaners collect the resources deposed in the transit area and move them to the waste area. Once a resource reaches the waste area, it is considered fully processed and will not be moved further. Workers carrying out the cleaning task are shown in blue.

Moreover, each time a resource is carried out by a worker and processed, it changes color and type. The resources outside the nest area (shown in green) switch from the foraging type to the dumped type. Resources in the dump area (shown in grey) switch from the dumped type to the transit type. Finally, resources in the transit area (shown in red) switch from the transit type to the waste type (shown in blue) once placed in the waste area.

[2]A uniform distribution is chosen so that the distribution of resources in the arena does not play a role in the performances of the different TAs

Furthermore, workers working on specific tasks will only recognize the resources of their current task – i.e., a forager will only be able to sense the resources of type foraging, and a robot carrying out the cleaning task will only be able to sense resources of the waste type.

This set of tasks (idle is omitted as it is more a state than a task) is a three dependent task system. It means that the foraging task's demand always influences the future need in the two others. In other words, for the demand to rise in the nest processing task, a resource first has to be collected by a forager and brought back home. For the demand to rise in the cleaning task, a resource first has to be collected outside and brought back home and then processed by the nest processors and moved to the transit area.

This three dependent task system can easily relate to real-life tasks such as collecting warehouse supplies and carrying them out to other parts of a hangar or in a transit area to be processed. **Note that** this task system is not representative or limiting the application of the AITA algorithm but only highlights the performances of this algorithm compared to other methods in this specific environment. The algorithm is expected to be as performant with a set of independent tasks.

4.4.1 Environment Assumptions

The current simulation environment yields the following assumptions; Firstly, it assumes that all the simulated agents are a set of homogeneous Thymio-II robots. Using a set of homogenous robots means that all the robots are the same, share the same capabilities and skills. Moreover, all the robots have the same navigation and object avoidance system. A robot can move somewhat precisely to a given coordinate in the plan during the simulation. At each new simulation, the robots are randomly distributed within the dump area, facing a random direction. They are assigned no task or no state. Additionally, 2000 resource items are distributed within the foraging area. The robots do not have any degrading variables, they remain as performant as they start throughout the simulation.

In the CAITA version, the communication has no overhead and can happen a soon as a robot enters the information center and opens a communication channel. The information center is assumed to be able to communicate back and forth with every robot simultaneously. As for CAITA, The DAITA system assumes an instant communication transmission time. However, each robot can only receive one transmission at a time (that is, one each simulation step), dumping any other incoming message up until the next simulation step. GTA and PSI, who use the DAITA communication system, suffer the same communication restrictions and challenges.

4.4.2 Environment Settings

Along with the environment's assumptions, a set of variables can be changed prior to running a simulation. It is expected that each of these variables has a high impact on the system's performance and results depending on its value:

- The number of robots
- The noise in/probability of communication failure
- The communication range
- The demand for each task
- The periodical increase of the demand in the foraging task

Periodical Increase of the Demand in the Foraging Task.

The periodical increase in the foraging task is a fixed value set ahead of all experiments. This parameter can be seen as irrelevant as the goal is to compare how fast a system can, for instance, collect N resources or how well does a system over a given period. However, the author argues that this periodical increase helps to keep the experiments as fair as possible for the following reason: Since the AITA algorithm makes sure that the demand for a task always meets equilibrium (given that enough workforce is available), it always tries to deploy as few robots as possible to cover it. Oppositely, systems such as RNDTA, GTA, and PSI always distribute all the workers over the task set, which creates a disbalance in the comparisons. By periodically increasing the demand, the environment makes sure AITA is as busy as possible, which means that the system is deploying all of its workforces to cover the different needs. Moreover, this variable pushes the different task allocation methods to show their performances in dynamic environments as the demand in the different tasks is constantly evolving.

Tests with 40 robots have proven that increasing the demand by 7 resources at every 500 simulation steps offers the fairest and most consistent results (an increase of 6 or less keeps the system in a lazy state with a significant part of the workers in the idle state).

5. EXPERIMENTS AND RESULTS

Table 5.1 describes the values used for the experiments. Each experiment is run five times and averaged.

Name	Value	Describes
R	Dependent of experiment	The number of robot used in the experiment
-	2000	The number of resources dispatched at the start of a simulation
C_{r}	See section *5.1*	Communication range
$Pnoise$	See section *5.1*	Probability of communication failure
-	500	Simulation step at wich the demand of the foraging task increases
$init_{\mathrm{np}}$	0	Initial demand of the nest processing task
$init_{\mathrm{c}}$	0	Initial demand of the cleaning task
$init_{\mathrm{f}}$	50	Initial demand of the foraging task
P	7	Value of the periodical increase in the foraging task
Θ	<R:50:7>	The relationship between R, $init_{\mathrm{f}}$, and P

Table 5.1: Experiment on AITA, variables initialization

5.1 Experiments on Parameters Settings

In order to best see the robustness and reliability of the CAITA and DAITA systems and choose variables that are the same for every task allocation method, tests upon communication failure/noise and change in the communication range have been conducted.

5.1.1 Experiment on Noise in Communication

The noise is implemented as a communication failure mechanism. Whenever a robot tries to broadcast its current knowledge of the world (DAITA) or report to the information center (CAITA), there is a probability $Pnoise$ that the communication with the receiver fails. The task is to fully process 150 resources with 40 robots as fast as possible. The system assumes that the communication range covers the entire arena. The tests are first run with a probability of communication failure $Pnoise$ of 0. Then, $Pnoise$ is gradually increased to reach $Pnoise = 0.99$.

DAITA Experiments

As shown in figure 5.1, the different levels of noise tested do not show any significant variation in the task completion rate (the rate at which the given task is completed).

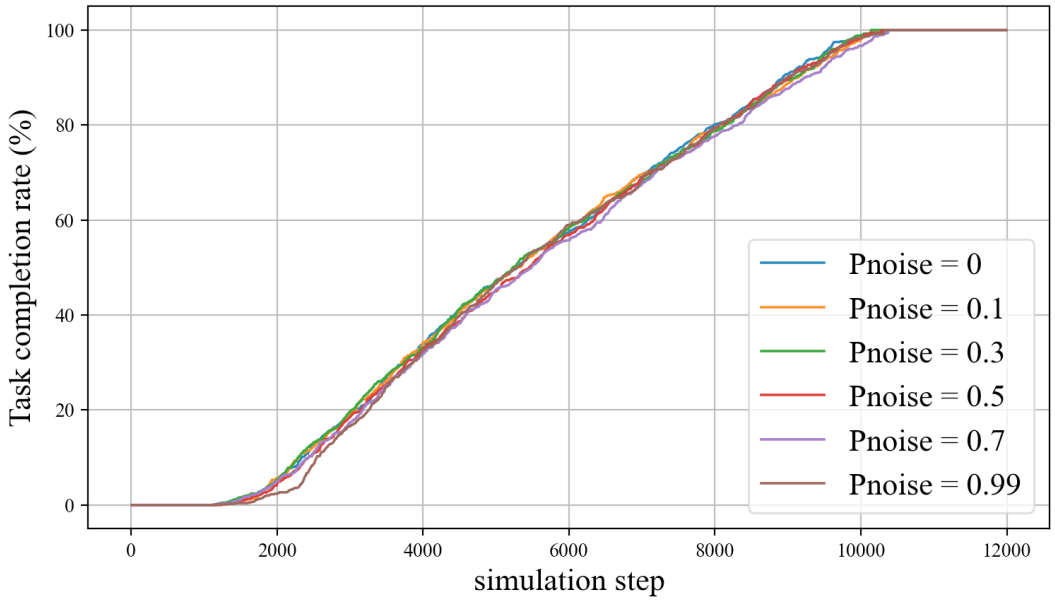


Figure 5.1: Experiments on Parameters Settings (Noise): Task completion rate, DAITA

These variations could mean that the system is highly robust even when 99% of the communications are lost. However, to further explore the noise's incidence on the system, one can look at the swarm's perception error E (Recall section *4.3 Investigation by Metric Evaluation*) of the actual environment.

Figure 5.2.a depicts the metric E for $Pnoise = 0, 0.3, 0.7$, and 0.99 where it can be seen that $Pnoise = 0$, $Pnoise = 0.3$, and $Pnoise = 0.7$ have the same tendency, reaching low fluctuation with an average error of 1.445 resources, 1.441 resources, and 1.48 resources respectively. Even at high noise levels such as $Pnoise = 0.7$, these slight variations are plausible and result from the communication system implemented in DAITA. In the DAITA communication system, each robot broadcasts its current knowledge of the world to all other individuals at each simulation step. In a system of 40 robots and a probability of success of 1 (that is, $Pnoise$ is set to 0), the number of successfully sent and received packets is; 1 * 39 = 39. Moreover, recall that each robot can only receive one packet per simulation step, which means that in this system, the robot has a likelihood of successfully receiving a packet equal to 100%. Now, in a system where the probability of success is 0.3 (that is, $Pnoise$ is set to 0.7), the number of successfully sent and received packets is; 0.3 * 39 = 11.7. The receiver (who can still only receive one packet per round) is probabilistically speaking, receiving 11.7 packets on average each round, more than 1, enough for the system to update and spread the information globally. Inversely, then the level of communication

failure reaches 0.99, the system cannot communicate fast enough and its error E reaches higher levels. This is because when the probability of success drops at 0.01 (that is, $Pnoise$ is set to 0.99), the robot is receiving on average 0.39 packets each simulation step, or less than one each round, which means that a successful communication can take up to 2.56 rounds to happen. Even though the tendency E for $Pnoise = 0.99$ skyrockets to a peak of 25 resources, the trends plummets, and the system stabilizes to reach around the same error E as the others, or 5.1 resources on average.

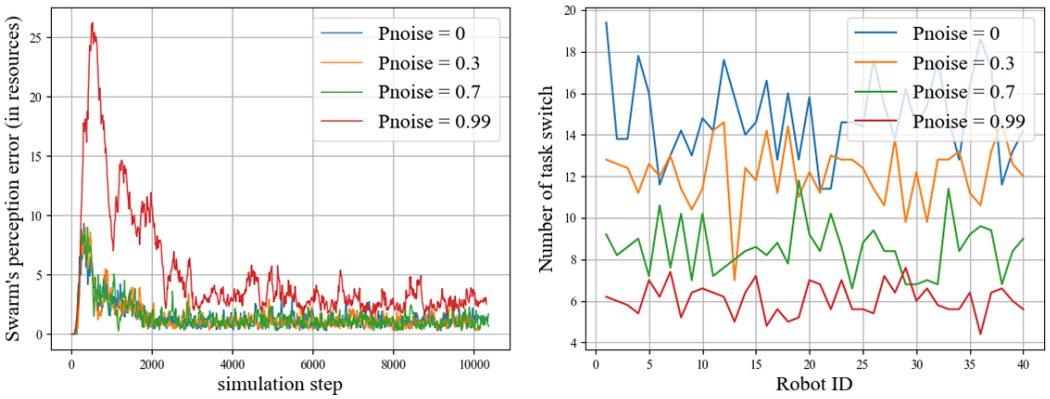


Figure 5.2: DAITA: a) Experiments on Parameters Settings (Noise): Swarm's perception error E. b) Experiments on Parameters Settings (Noise): number of task switches per robot

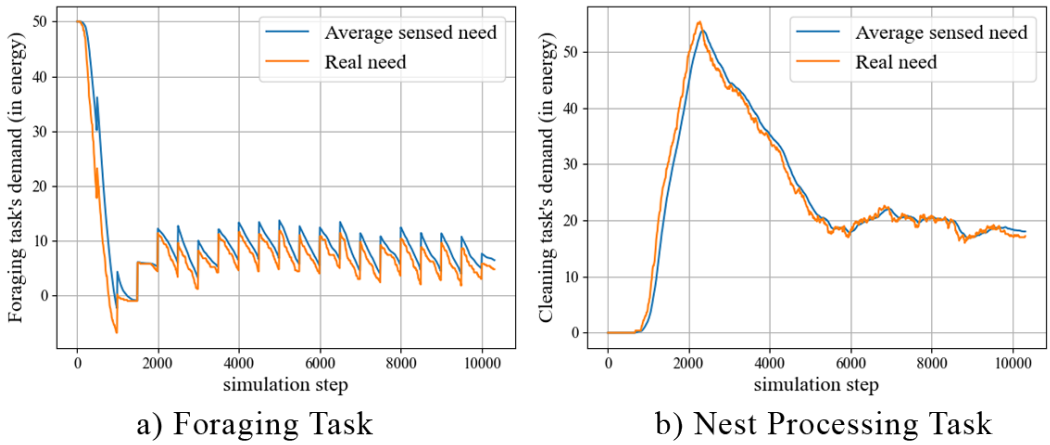


Figure 5.3: Experiments on Parameters Settings (Noise: 0.99): Tasks needs - Sensed and Real, DAITA

As the probability of communication failure reaches a high level, the success rate of communications can become so low that the swarm struggles to adapt to the current state of the environment. This tendency of difference between the status of the environment and what is seen by the swarm can further be seen in figure 5.2.b where as the noise is low (for

$Pnoise$ = 0 and 0.3), the current knowledge of the world is high (as information is shared faster) resulting in the robots adapting faster to the environment. Fast adaptation comes with high oscillations in the task allocation, resulting in a high number of task switches per robot. Inversely, for a high probability of communication failure, the swarm struggles to quickly adapt to the changing environment as failure in communication means the information is shared at a slower pace, which results in a low oscillation in the distribution of labor, leading to a lower number of task switches per robot.

The consequence of delay in information sharing caused by a high probability of communication failure, also called the *knowledge error shift*, can further be seen in figure 5.3 (graphs of the average sensed need and the real need for each task, $Pnoise$ = 0.99) where, as the demand for resource collecting gently drops, the average sensed demand (in blue) is slightly shifted compared to the actual demand (in yellow). Moreover, as the information is shifted, the workers do not have the information of the tasks being fulfilled in time. Not getting tasks' information in time is a sign of faulty labor distribution, leading to possible over-collection of resources (when a forager keeps collecting resources although the demand is met). Over-collection of resources explains why the curve slightly variates for $Pnoise$ = 0.99 in figure 5.1 and has a similar task completion rate to lower noise levels. Indeed, over-collecting compensates for the periodical increase in the foraging task, which over short periods of time gives an advantage[1]. However, faulty and inconsistent distribution of labor proves that high levels of noise are harmful to the system.

CAITA Experiments

In the case of CAITA, the task completion rate shown in figure 5.4 demonstrates signs of weakness when the level of noise in the system reaches 0.99, finishing the task with 54270 simulation steps. Oppositely, the other levels of noise achieve better performances, reaching 100% with 10771, 10952, 11028, 11125, and 11599 simulation steps for $Pnoise$ = 0, 0.1, 0.3, 0.5, and 0.7, respectively. To understand why there is a significant difference between the results of CAITA and DAITA for $Pnoise$ = 0.99, one can look at the distribution of labor shown in figure 5.5. It can be observed from the graph that a high level of communication failure such as $Pnoise$ = 0.99 generates inconsistent labor distribution (in opposition to lower levels) resulting in the system deploying only about half of its worker, even though it is supposed to be kept highly busy for the entire simulation period (recall section *4.4.2 - Periodical Increase of the Demand in the Foraging Task*). Why is the distribution of labor so inconsistent? Recall that the communication system of CAITA is fundamentally different from the one implemented for DAITA. In CAITA, each worker has to report to a central, which then distributes tasks. Back and forth communication means that the likelihood of a failed communication is doubled, thereby amplifying the effects of noise. Thus, the probability of receiving a new task assignment is reduced, resulting in a slow distribution of labor.

[1] Under the specific set of assumptions, tasks, and environments defined for this thesis.

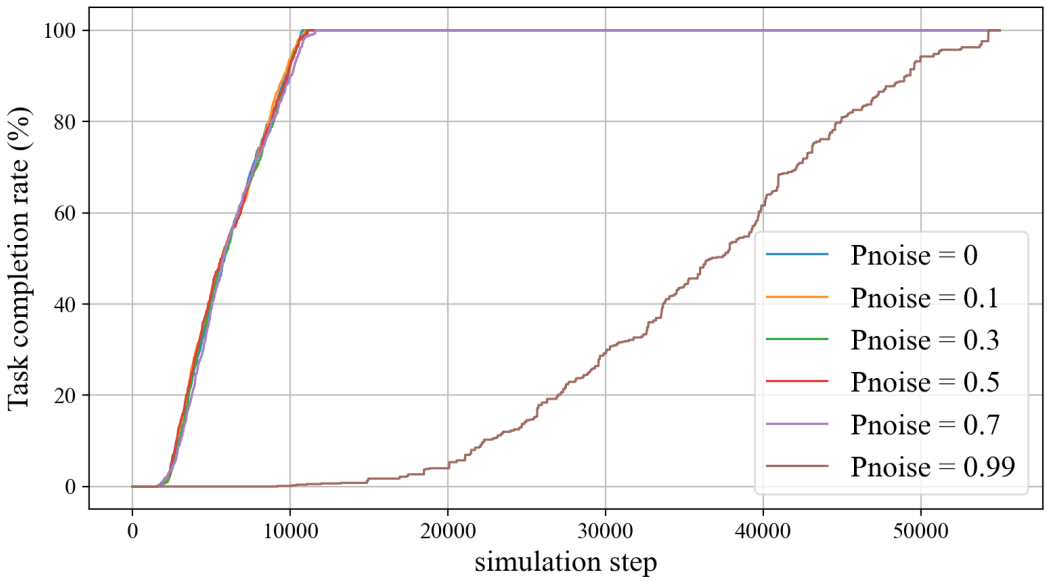


Figure 5.4: Experiments on Parameters Settings (Noise): Task completion rate, CAITA

When looking at the nest's perception error E for $Pnoise = 0, 0.3$, and 0.7 shown in figure 5.6, the effects of faulty communication get less obvious compared to the one observed for DAITA (figure 5.2.a). Indeed, recall section *3.6.2 Integration: Centralized Ant-Inspired Task allocation*; In the CAITA communication system, it might happen that a robot successfully sends its information to the information center (thereby updating the table of global knowledge) but fails to receive the new task assignment back from the information center. As the nest's perception error E is a measure of the need sensed by the nest and the real need, the failed reception from the receiver robot does not affect it. Nevertheless, the nest's perception of the CAITA system is more prone to errors compared to the swarm's perception of DAITA as the robot only has one chance to send out a successful report per simulation step; when it reports to the nest. To reflect on the probabilistic calculation done in the previous section, the same types of calculus are conducted; when the success rate of communication is set to 0.03 (that is, $Pnoise$ is set to 0.7), there is a 0.7 * 1 (1 = one receiver, the information center) chance that the communication is successful.

Consequently, there is a 0.01 * 1 chance of successful communication when $Pnoise$ is set to 0.99, explaining how E reaches an average of 21.24 resources over the simulation period in figure **??**. The effect of the nest's perception error can further be seen in figure 5.7. The graphs show the real needs (in yellow) and the need sensed by the information center (in blue) for $Pnoise = 0.99$. As the simulation runs, the effects of shift due to communication failures are more visible.

In conclusion, both DAITA and CAITA show a high level of robustness to commu-

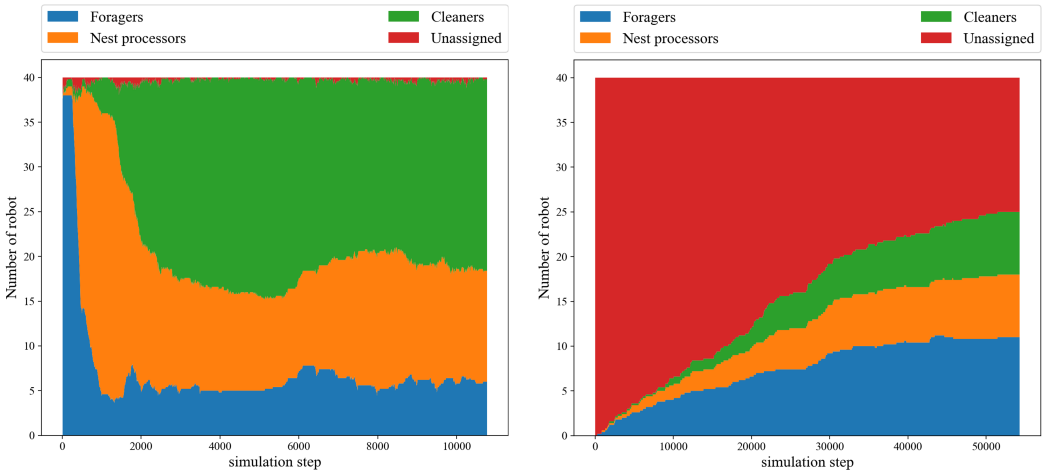


Figure 5.5: Experiments on Parameters Settings (Noise): Distribution of labor for $Pnoise$ = 0, 0.3, 0.7, 0.99, CAITA

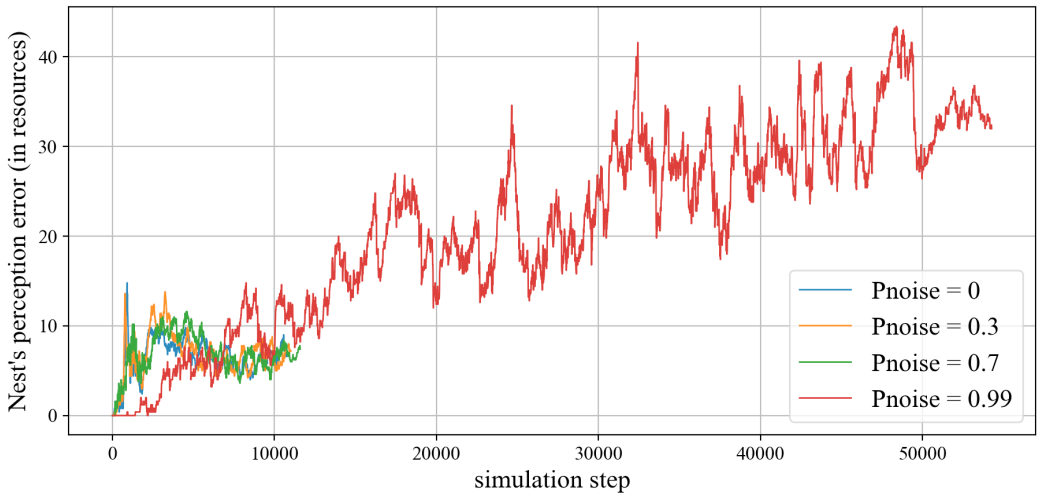
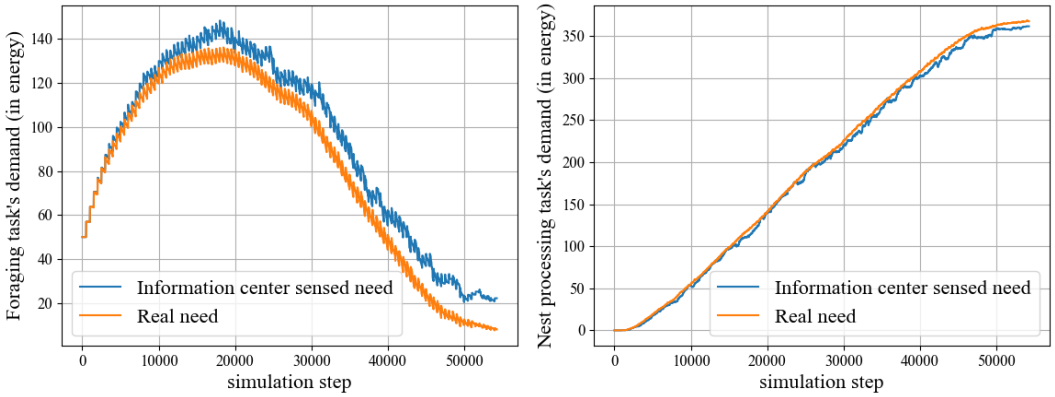


Figure 5.6: Experiments on Parameters Settings (Noise): Nest's perception error E , CAITA

nication failures for noise levels reaching up to 70%. However, both systems demonstrate weakness as the noise level reaches 99% and slows down communication. CAITA suffers the most from this noise level as back and forth communication worsens communication failure effects. Moreover, it does not show any recovery over an extended period as can be observed in figure 5.6, where the $Pnoise$ = 0.99 curve does not fall down. Oppositely, DAITA shows signs of recovery as its trend peaks down (figure 5.2.a) to reach an error close to lower levels of communication failure such a $Pnoise$ = 0.3. In a highly dynamic environment, it can be expected that a system with a high noise level, such as 70%, will perform poorly and will not be able to adapt to the changing situation fast enough as its knowledge of the world is faulty. Experiments and reflections of poor performances due to



a) Foraging Task b) Nest Processing Task

Figure 5.7: Experiments on Parameters Settings (Noise: 0.99): Tasks needs - Sensed and Real, CAITA

a high noise level are not covered in this thesis. However, the Introduction section in the work of Alvarez-Aguirre et al. [27] provides a good understanding of the problem.

For the rest of the experiments, the level of noise in communication failure *Pnoise* is set to 0.3. 30% of noise offers consistent results and a fast adaptation to the environment while remaining a challenging level of communication failure.

5.1.2 Experiment on Communication Range

As the robots use radio-telecommunication devices to communicate on the world's current status and to share their current progress (Recall section *3.6.3 Integration: Distributed Ant-Inspired Task allocation*), experiments on the effect of different lengths of communication range are relevant to this thesis. The experiments assume that a communication device such as a Zigbee wireless module which can communicate up to 100m [28], is placed on top of the agents to enable multidirectional short-range/long-range communication (multidirectional - whether the robot receiving the signal is in front or behind does not matter as long as it is within the range). The experiments are conducted on DAITA[2]. The swarm has to collect and fully process 150 resources with 40 robots. The range is first of 0.1 meters and incrementally increases to 13 meters, covering the entire arena. The task completion rate results for the ranges 0.1m, 0.5m, 1m, 5m, 7m, and 13m are shown in figure 5.8. Although the previous section has set the level of noise to 30% for the rest of the experiments, tests on communication ranges assume no noise in order to provide the best investigation.

As can be observed, the task completion rate for the range 0.1 meters is the worst, finishing its task some 20'000 simulation step later than any other ranges. Oppositely, communication ranges of 5, 7, and 13 meters have similar trends and offer the most consistent

[2]Recall that CAITA has no range. Instead, it assumes a worker can communicate with the information center as soon as it enters the communication area.

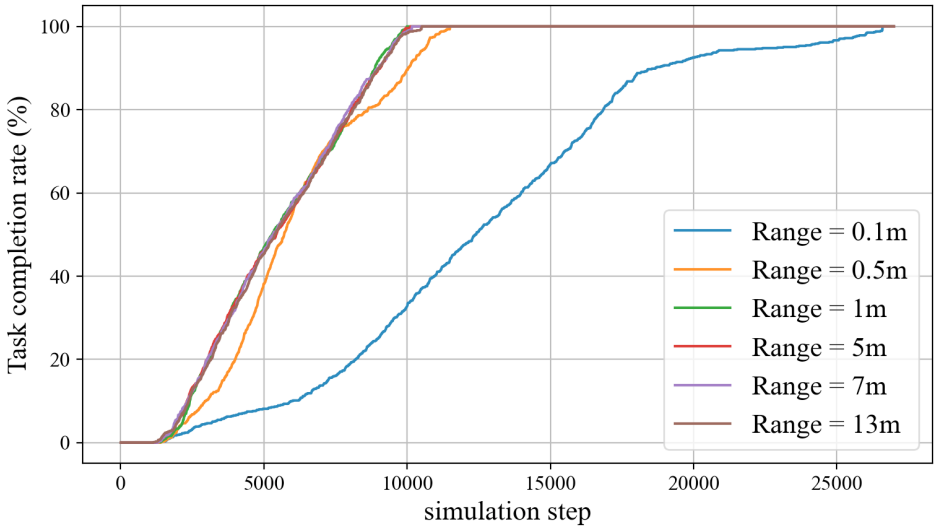


Figure 5.8: Experiments on Parameters Settings (Range): Task completion rate, DAITA

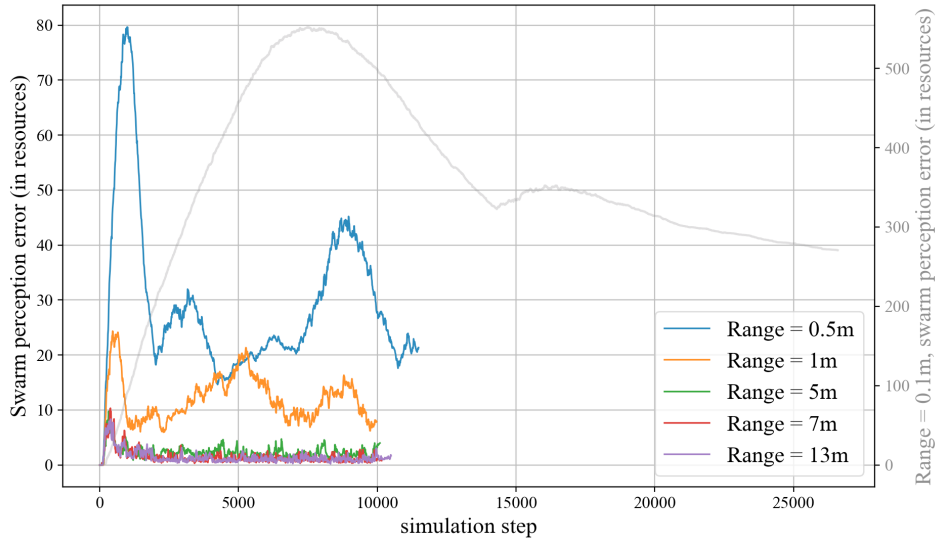


Figure 5.9: Experiments on Parameters Settings (Range): Swarm's perception error E, DAITA

The 0.1m curve is shown in grey. Its scale is on the right as it is significantly different.

result and a high task completion rate. Finally, the trends of communication ranges 0.5 and 1 meter show slight variations as they complete their assignment in a non-linear fashion compared to other communication ranges. As for the section *5.1.1 Experiment on Noise in Communication*, the task completion rate is not enough to demonstrate the effect of shorter communication ranges over the DAITA system. The swarm's perception error E of the actual environment is recorded and shown in figure 5.9 to highlight possible issues with shorter ranges.

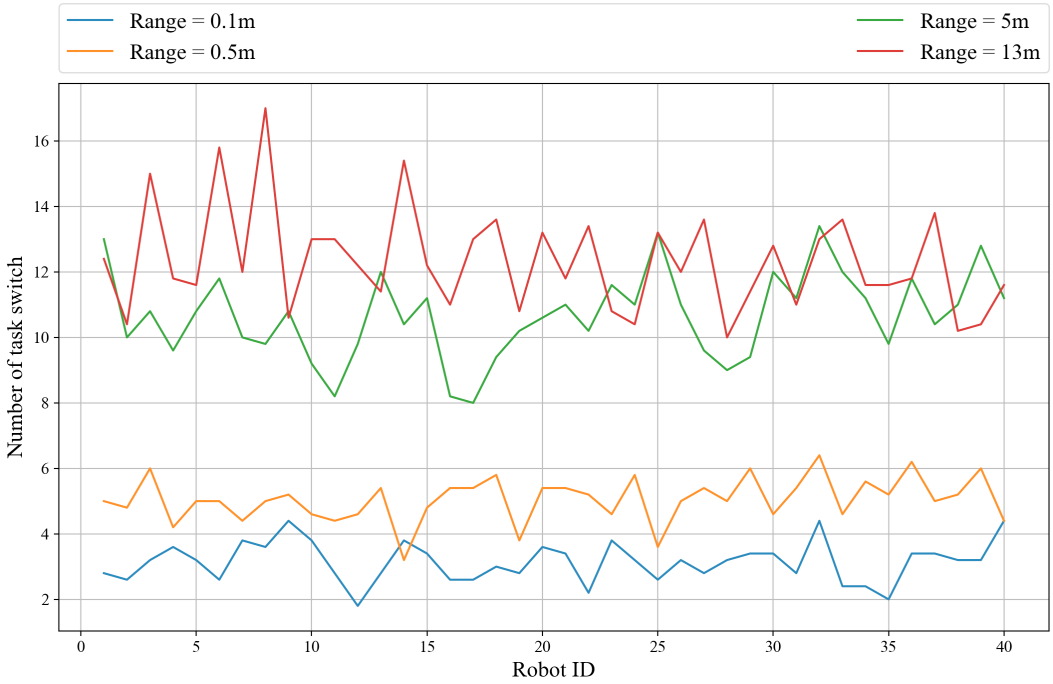


Figure 5.10: Experiments on Parameters Settings (Range): Number of task switches per robot, DAITA

Figure 5.9 shows that communication ranges of 0.1, 0.5, and 1 meter offer the lowest speed in global information sharing, reaching an average error E of 352.92 resources, 29.49 resources, and 11.95 resources, respectively. Conversely, communication ranges of 5, 7, and 13 meters offer a low level of error, reaching an average error E of 2.31 resources, 1.47 resources, and 1.43 resources, respectively. Although the effects of short communication ranges are consequent on the system, recovery can be observed over time as their curves have a downward trend (see curves for ranges 0.1, 0.5, and 1 meter in figure 5.9).

Harmful effects due to short communication ranges can be seen in the number of task switches per robot shown in figure 5.10 (number of task switches for ranges 0.1, 0.5, 5, and 13 meters). In systems with a short communication range, such as 0.1 or 0.5 meters, global information sharing is delayed, resulting in fewer task switches. In contrast, in systems with long-range communication, such as 5 or 13 meters, the global information is shared faster, resulting in a higher number of task switches per robot.

Short-range communication's effects can be correlated to the knowledge error shift discussed in section *5.1.1 Experiment on Noise in Communication*. Recall that the knowledge error shift appears as the communication is made slower and the knowledge about the current world situation struggles to be shared within the swarm. The graph series in figure 5.11 shows the real need and the sensed need (averaged) by all robots in the environment for ranges of 0.1, 1, 5, and 13 meters. In the 0.1 meters graph, it can be seen that the average sensed demand of all robots gets significantly more shifted compared to the real need as

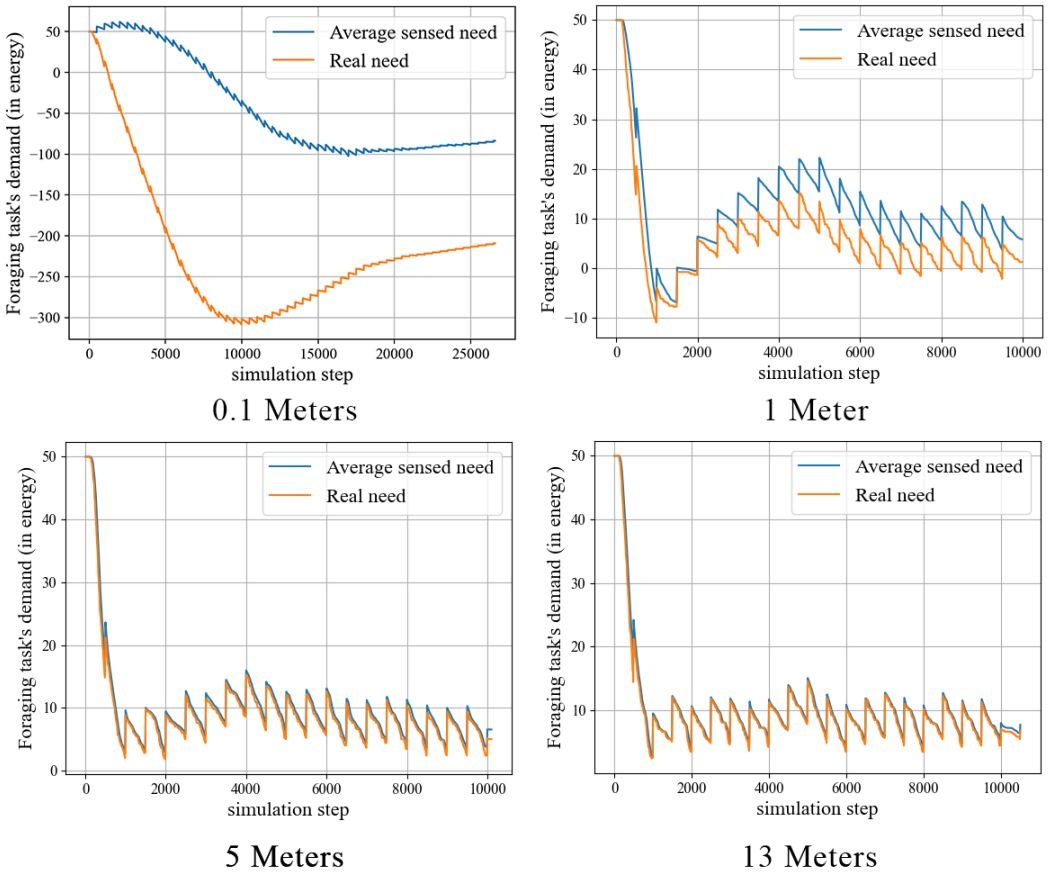


Figure 5.11: Experiments on Parameters Settings (Range): Foraging tasks needs - Sensed and Real, DAITA

the simulation goes, resulting in over-collection. As the communication range gets longer (graphs for 1, 5, and 13 meters), the shift gets less significant, and the over-collection effect fades out.

A solid feature of the environment related to a healthy system is the visible periodical peaks in the distribution of labor, as seen in the series of graphs in figure 5.12. These periodical peaks happen as the demand in the foraging task periodically increases, signifying that the global information is successfully shared throughout the swarm. The first two graphs are the labor distribution of the range 0.1 meters and 1 meter; no peaks are visible. Inversely, the two other graphs, distribution of labor for range 5 and 13 meters, show signs of swarm adaptation as the periodical peaks are visible throughout the simulation period.

In conclusion, the system shows solid results to different communication ranges. However, short communication ranges such as 0.1, 0.5, and 1 meter result in communication delay preventing the swarm from adapting fast enough to the demand, considerably increasing the swarm's perception error E.

For the rest of the experiments, the communication range is set to cover the entire

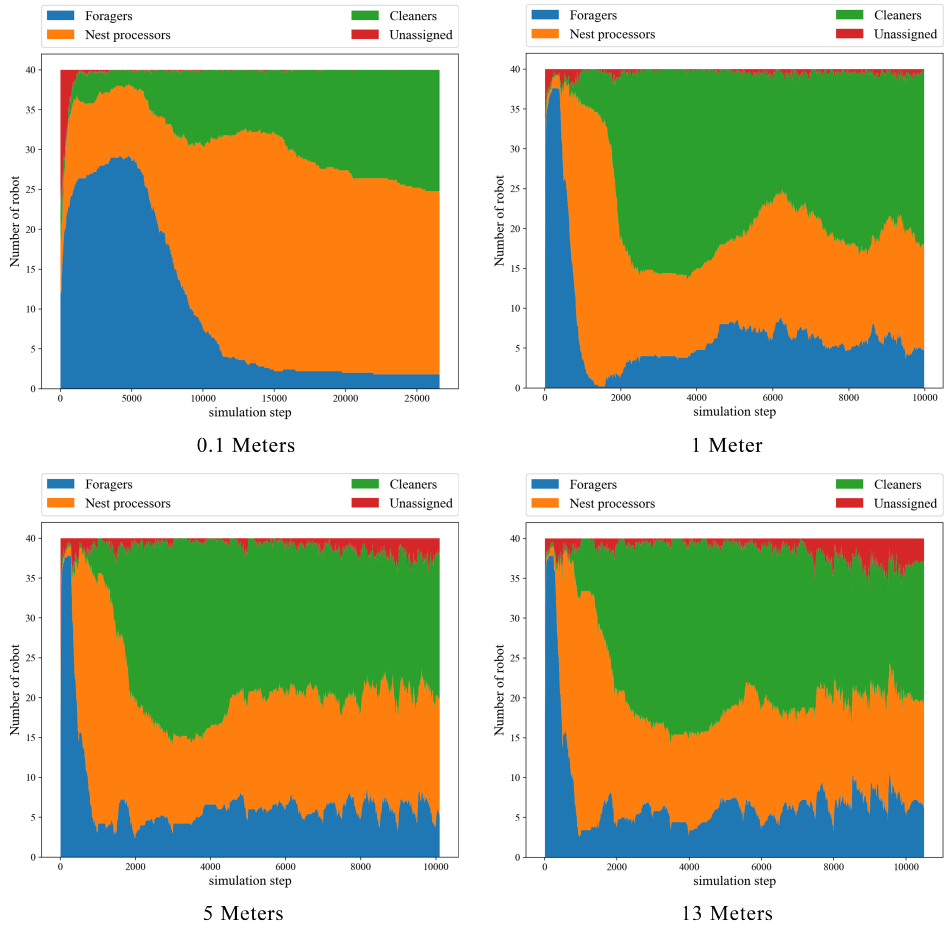


Figure 5.12: Experiments on Parameters Settings (Range): Distribution of labor for C_r = 0.1, 1, 5 and , 13m, DAITA

arena (13 meters). Although communication ranges of 5 and 7 meters offer high consistency while reducing the swarm's perception error, the results also underline that the longer the communication range gets, the less significant the improvements are. Thus, and for simplicity, 13 meters is chosen as it will offer the highest consistency and accuracy in the results of the experiments.

5.2 Task Completion Rate

The task completion rate is meant to observe the speed at which a group of individuals can complete a given task. Moreover, the task completion rate can be used as an optimization goal as the sooner the swarm completes its task, the better the system is. In order to experiment with the optimization metric, experiment 1 is defined. It consists of collecting, processing, and cleaning 150 resources as fast as possible[3].

[3]Simulation time limit has been set to 40'000 simulation steps as some experiments can take over 200'000 simulation steps to complete, irrelevant for comparison.

5.2.1 Task Completion Rate on Variating Numbers of Robots

The first set of tests performed on the CAITA and DAITA systems with the number of robots R variating from 10, 20, 30, 40, 50, 70 to 100 shown in figure 5.13.a and 5.13.b, show that the task completion rate with 40 and 50 robots is significantly higher than for 10, 20, and 30 robots. This trend shows a direct relationship between the number of robots performing a task and the completion rate of a task. However, the task completion rate trend is not a linear function to the number of robots as both CAITA and DAITA demonstrate a lower task completion rate when this number goes above 50. Indeed, in the former, using 70 robots or 100 robots is barely as good as using 50 robots to achieve the same performance. In the latter, using 50 robots and 40 robots remain the overall fastest as 70 robots, and 100 robots obtain a lower task completion rate.

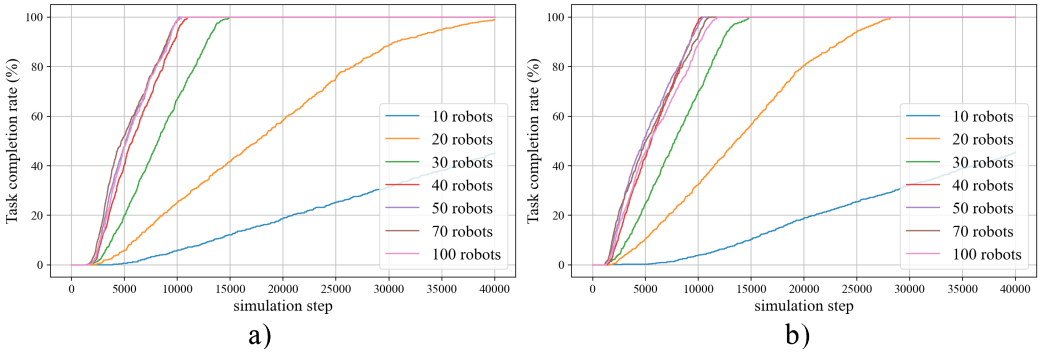


Figure 5.13: Experiments on Task Completion Rate: Variating number of robots for a) CAITA and b) DAITA

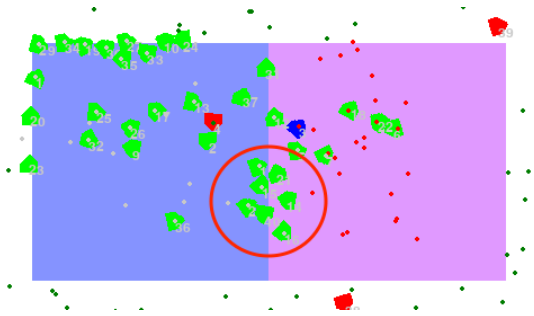


Figure 5.14: Robot congestion in simulation

Mild congestion of the system. As more robots join the same task, more of them need to gather in the same area, creating a congestion in the nearby surroundings.

Many variables can play a role in the drop of performances observed when reaching higher numbers of robots. For instance, robot congestion (as shown in figure 5.14) – i.e., when multiple robots try to reach a similar goal and struggle to find their way through as others block them. Furthermore, congestion implies that optimizations of the task completion rate

by improving the positioning of the different areas and the implemented robot-avoidance algorithm are possible, however not covered in the experiments.

Another part belongs to how AITA has been designed to deal with this type of situation. As explained previously, the algorithm intends to deploy as few robots as needed on a task to reach equilibrium. While deploying more robots might imply better performance, in an environment of 70 or 100 robots, AITA will only deploy as many robots as needed to cover all tasks needs. This leads to a significant part of the colony to remain idle if the demand is not high enough to occupy 70 or 100 robots, creating more congestion as idle robots gathered in the nest prevent the working robots from reaching their endpoints. This behavior can be considered as "underperformance", but it is just the AITA algorithm not using more energy than needed to complete a given task. The way AITA deploys its workforce to cover the needs of the tasks underlines a direct relationship between the number of robots and how busy the system is. This relationship is elaborated on in the following subsection.

Relationship Between the Number of Robots and the Initial Demand of the Foraging Task

The relationship between the number of robots and the initial demand of the foraging task can be written as follows: $< R : init_{\mathrm{f}} >$, or Θ. This relationship is tightly bonded to the fixed periodical increase in the demand of the foraging task, P, described in the section *4.4.2 - Periodical Increase of the Demand in the Foraging Task*. The relation can then be rewritten as follows: $\Theta =< R : init_{\mathrm{f}} : P >$, or the *system's business*, where P is a fixed value. If no significant improvements in the task completion rate are observed when using 70 robots or 100 robots, it is due to a configuration of Θ where the workload of the environment is low, keeping a large population of individuals idle, as shown in figure 5.15.a ($\Theta =< 100 : 50 : 7 >$). Inversely, the distribution of labor shown in figure 5.15.b demonstrates a system where $\Theta(< 40 : 50 : 7 >)$ is configured to obtain a high workload, keeping the system busy over the simulation.

To further observe the effect of different Θ settings on the optimization of the task completion rate, figure 5.16 shows the results of experiment 1 (using DAITA) conducted with the following Θ parameters: $< 100 : 25 - 50 - 75 - 100 - 150 - 200 : 7 >$. As can be observed, the choice of different values for Θ highly influences the result obtained. The trends show an optimization of the task completion rate as $init_f$ grows. However, the performances are weakened when $init_f$ is set to a value higher than 100 as it takes more time to collect 150 resources since the agents' workforce is focused on the foraging task. Therefore it is crucial to select a value of Θ that reproduces the *system's business* of the other investigated algorithm in order to provide a fair comparison. In this regard, thorough experiments (not shown in this thesis) on the Θ parameter have highlighted that $\Theta =< 40 : 50 : 7 >$ is best to reproduce the desired *system's business*.

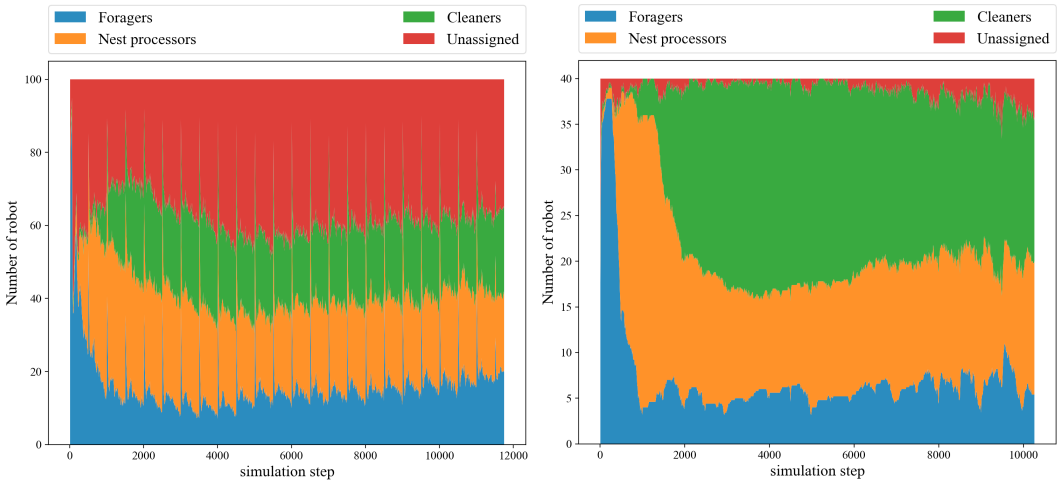


Figure 5.15: Experiment on parameter Θ. a) Θ =< 100 : 50 : 7 >. b) Θ =< 40 : 50 : 7 >

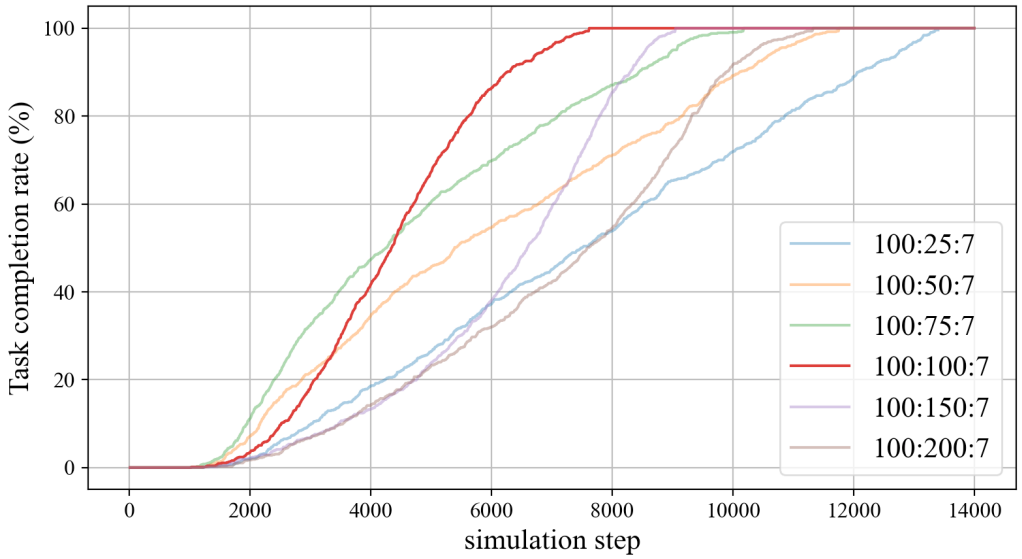


Figure 5.16: Experiment on parameter Θ

5.2.2 Task Completion Rate On Different Task Allocation Methods

The same experiment is conducted on RNDTA, GTA, and PSI with a parameter Θ set to < 40 : 50 : 7 > and compared to the two previous systems. In figure 5.17, the task completion rate does not show significant variation between GTA (100% at 10024 simulation step), DAITA (100% at 10262 simulation step), and PSI (100% at 10389 simulation step). However, the task completion rate of CAITA is significantly lower than its peer (100% at 10935 simulation step), indicating a better performance for the distributed architecture. Nonetheless, CAITA still achieves better performances than RNDTA, which achieves the worse task completion rate (100% at 12536 simulation step).

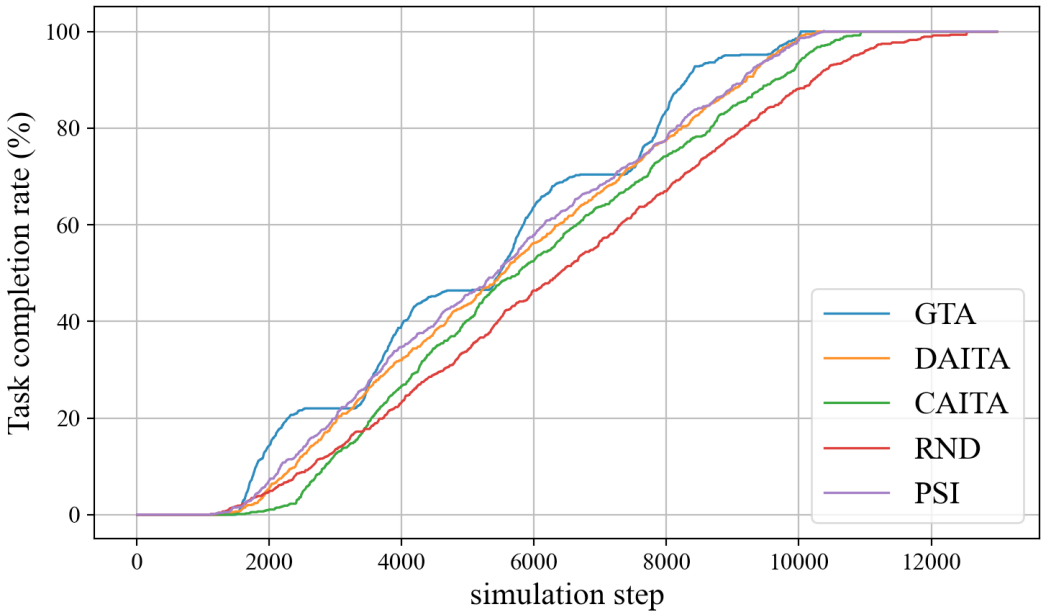


Figure 5.17: Task completion rate on different task allocation methods

In conclusion, the results of the task completion rate experiments yield that there is a fundamental relationship expressed by the parameter Θ that drastically changes the performances of AITA. The *system's business* is introduced and describes how Θ can be shaped to enable replications of other task allocation methods' performances that do not work like AITA (as AITA does not distribute all of its workers if the demand is met). As a result, when compared to other systems, AITA shows high performances and fast completion time, although the centralized version performs slightly worse.

The task completion rate is tightly bounded to how well a system can distribute its workers among the different tasks. The better the system can optimize it, the faster it completes its assignment. The following section provides an investigation of which system best optimize this distribution.

5.3 Optimization of the Labor Distribution on Different Task Allocation Methods

Cornejo et al. define an optimal task assignment as one that reduces the squared difference between the need for a task and the energy applied to it. To prove that their system can reach near-optimal distribution of labor, Experiment 2 is defined. Experiment 2 ($\Theta =< 40 : 50 : 7 >$) is a long-run simulation of 30'000 simulation steps (no goal).

Figure 5.18 shows the labor distribution optimization metric for the different task allocation methods. It can be observed that methods that consider the demands in their task allocation algorithm do highly better than those that do not, such as RNDTA. RNDTA

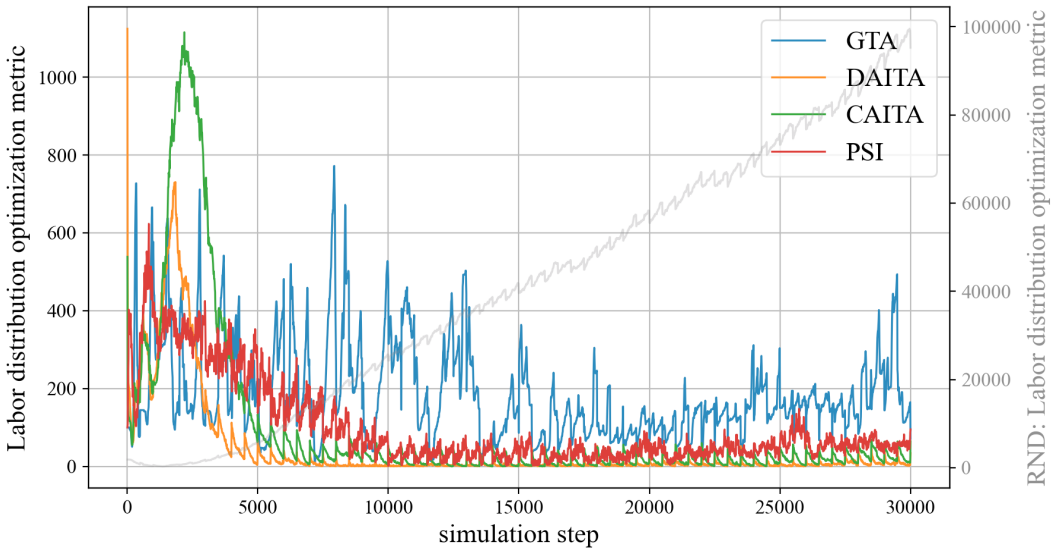


Figure 5.18: Labor distribution optimization metric of different task allocation methods

RNDTA's curve is shown in grey. Its scale is on the right as it is significantly bigger.

reaches an average of 41484 squared difference, and the upward trend shows that this number is not falling soon. As RNDTA distributes its workforce randomly over the three tasks instead of considering the demand, it over-collects. As it over-collects, the demand in each task is not met, and the growth never stops. Oppositely, PSI, CAITA, and DAITA, with an average squared difference of 99, 96, and 44, show great performances as their trend tends towards zero over time, reaching near-optimal task allocation. GTA shows the same downward trend as DAITA, CAITA, and PSI. However, its squared difference over the same period is of 186 on average, which underlines high inconsistency in the distribution of labor, far from reaching a near-optimal curve.

In conclusion, both CAITA and DAITA offer the highest labor distribution optimization. However, CAITA performs slightly worse than DAITA due to communication delays in the centralized information system. Indeed, the labor distribution optimization is limited by the time it takes for information about a task's progress to be shared throughout the swarm. In CAITA, the system takes longer to share this information as the robot only contacts the information center if they have to or are in its coverage area. Oppositely, the data flows faster in DAITA as robots continually share information regardless of space and time.

Nevertheless, the results obtained in this section prove that over time, AITA reaches a near-optimal labor distribution.

5.4 Optimization of the Energy Consumption

The energy consumption of the system is investigated based on experiment 2. Two optimization goals are defined; The optimization of the system's energy consumption in terms of 1) the total covered distance by all robots and 2) in terms of the number of task switches per robot.

5.4.1 A Comparison of the Total Distance Traveled by Robots on Different Task Allocation Methods

The total distance covered by the entire swarm reflects the energy consumption of the different task allocation methods. The longer the traveled distance is, the more energy a system has consumed over its lifetime.

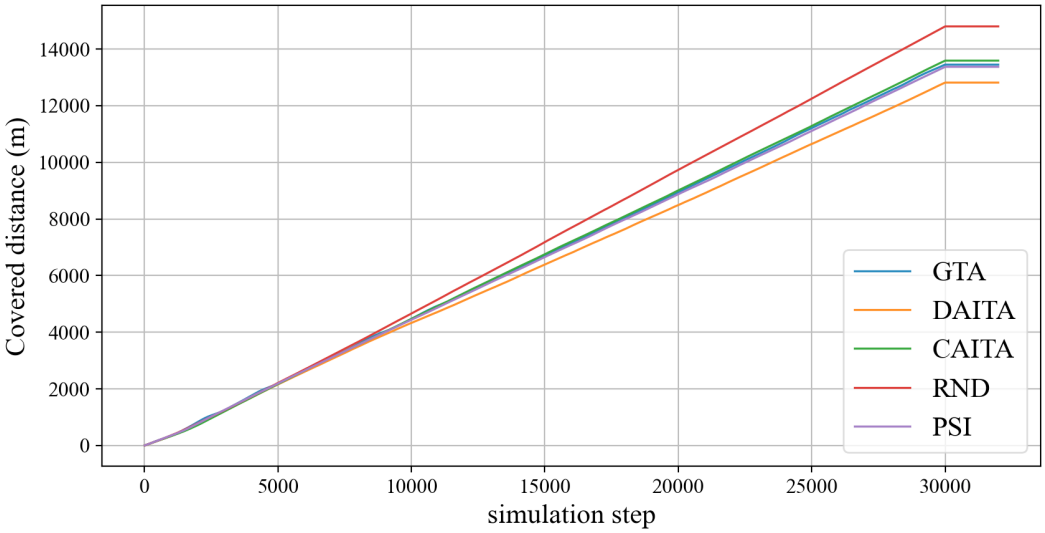


Figure 5.19: Total covered distance by all robots for different task allocation methods

Figure 5.19 shows that over a given period, DAITA minimizes the energy used, covering a distance of 12816m, 557m less than the second to best in energy consumption, PSI, with a covered distance of 13373m. GTA and CAITA cover almost the same distance as PSI with 13451m and 13591m, respectively. In comparison, RNDTA has a high energy consumption with a total covered distance of 14801m. The reason is that the algorithm deploys robots over tasks that sometimes do not require attention as their demand is already met, generating more movement than necessary. Oppositely, if DAITA achieves great performance, it is because its core algorithm has been designed to optimize the number of workers allocated to a task as a function of its demand. In most cases, this means that if all tasks are already in equilibrium, task-less robots will remain idle, effectively not moving and consuming no energy. Unlike DAITA, workers in CAITA have to report their status to the information center periodically. It means that there is a likelihood that the energy demand

for a task is met while another robot is outside the coverage area of the information center, thereby using slightly more energy as it could have stopped moving if the information would have reached it sooner.

Overall, both CAITA and DAITA show great performances in energy consumption in relation to the covered distance. DAITA optimizes the best its energy consumption compared to CAITA who is consuming more or less the same amount of energy as PSI or GTA.

5.4.2 A Comparison of the Number of Task Switches Per Robot on Different Task Allocation Methods

By minimizing the number of task switches per robot, the task allocation methods reduce the system's disturbance, resulting in less energy used over the simulation period as robots move less frenetically within the different areas. Figure 5.20 shows the number of task switches per robot for each system. It can be seen that PSI has the highest number of task switches per robot compared to the other task allocation methods, with an average of 1774[4]. CAITA is the one that maximizes the optimization variables the best, reaching on average

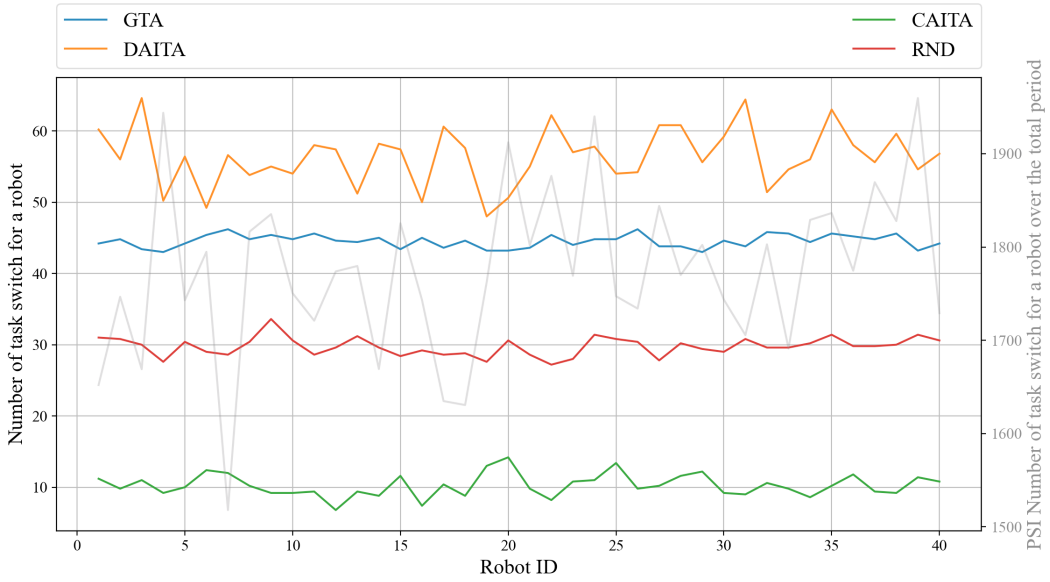


Figure 5.20: Number of task switches per robot for different task allocation methods

PSI's curve is shown in grey. Its scale is on the right as it is significantly bigger.

29 task switches per robot. RNDTA and GTA obtain better results than DAITA (with 56 task switches per robot on average), scoring 29 and 44 task switches per robot on average. However, the comparison is hard to make between DAITA, GTA, and RNDTA as both

[4]Recall that PSI suffers from the difference between its original environment and the one used for this thesis.

GTA and RNDTA are deterministic algorithms. This time, the centralized task assignment distribution coupled with the periodical report of the robots' status has given an advantage to CAITA as workers ask for a new task assignment less often than for DAITA. Moreover, CAITA's curve fluctuates less than DAITA's, meaning that its distribution is more consistent.

In conclusion, AITA shows a great performance in energy consumption in relation to the number of task switches per robot compared to other methods. However, the distributed architecture shows an inferior optimization due to its communication system more prone to oscillations, causing more task switches over the simulation period.

5.5 Adaptive Change in the Workforce

The adaptive change in the workforce is meant to observe how well a group of individuals can adapt to a sudden change in the number of workers performing a task. The adaptive change in the workforce can also be used as an optimization goal as changing variables of the environments such as the communication efficiency, the robot's efficiency at task solving, the number of robots, and more can influence it. Experiment 3 is defined to observe the effect of the adaptive change of the workforce by testing the scalability, robustness, and adaptability of the system. The experiment consists of removing the class of workers performing the nest processing task at a given simulation step and re-introducing it later on. The robots are removed at the 20000th simulation step, which gives the system enough time to start and stabilize itself. The robots are then re-introduced at the 40000th simulation step, and the simulation stops at the 60000th simulation step. During the time the class of workers is gone, and soon after the class is re-introduced, adaptability in the current environment is expected to be observed. The experiment is run with $\Theta =< 40 : 50 : 7 >$.

Note that: In GTA, each robot simultaneously switches to the task with the highest demand (though it may happen that, due to communication delay, one robot switches later than its neighbors). It means that there is a 2/3 likelihood that no robots are removed from the simulation when the 20'000th simulation step comes. Therefore, GTA is removed from this experiment as it might prevent any relevant data from being generated.

Figure 5.21 shows the labor distribution optimization metric for CAITA, DAITA, PSI, and RND. It can be observed that all methods suffer from the change in the workforce at the 20000th simulation step as their trends skyrocket due to the lack of demand coverage[5]. However, when the robots are re-introduced, signs of recovery appear as CAITA, DAITA, and PSI's trends reach lower levels. Moreover, based on their current trend, it can be

[5]If RNDTA does not seem to be negatively impacted by the change in workforces, it is because its optimization of the labor distribution to cover the relative demand of the swarm is already inefficient. Thus, when robots are removed from its system, it reduces the inefficacity effects as fewer robots over-collects.

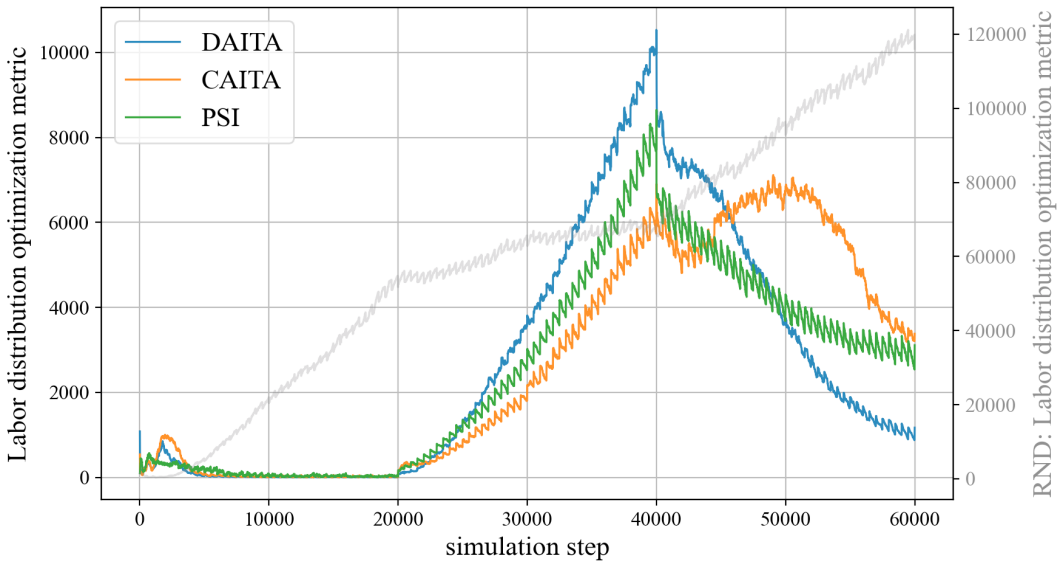


Figure 5.21: Adaptivity in the change of workforce; Labor distribution optimization - DAITA, CAITA, PSI, and RND

RNDTA's curve is shown in grey. Its scale is on the right as it is significantly bigger.

expected that given enough time, the methods go back to a more optimal task assignment.

Although DAITA displays the worse labor distribution optimization compared to CAITA and PSI when the robots are removed, but offers the best recovery when the robots are re-introduced. This fast recovery might result from DAITA's fast communication system providing a quick adaptation from the swarm to the newcomers. Inversely to DAITA, CAITA shows the best labor distribution optimization when the robots are removed, but demonstrates the worse optimization when the robots are introduced.

AITA's adaptability in the change of workforce can further be seen in figure 5.22 (Distribution of labor for a) CAITA and b) DAITA). The labor distribution graphs show that both methods adapt to the loss of workforce by distributing robots from other tasks in the nest processing task. As the group of workers is re-introduced, they are once again dispatched proportionally over the set of tasks[6], proving high adaptability in the change of workforce for both methods.

It can be concluded that both CAITA and DAITA show great strength and robustness to a sudden change in their workforce. However, the methods display a mirrored behavior of strength and weakness as the robots are removed or re-introduced, highlighting

[6] At the 40'000th simulation step, it might look like the group of workers is also re-introduced in the foraging task although it is only supposed to be re-introduced as removed, that is, in the nest-processing task. If this looks like it, it is because once re-introduced, the robot re-joins the nest processing task, which energy demand has already been met by other robots. This leads the newly re-introduced robot to switch to other tasks in energy demand, such as, in this case, the foraging task.

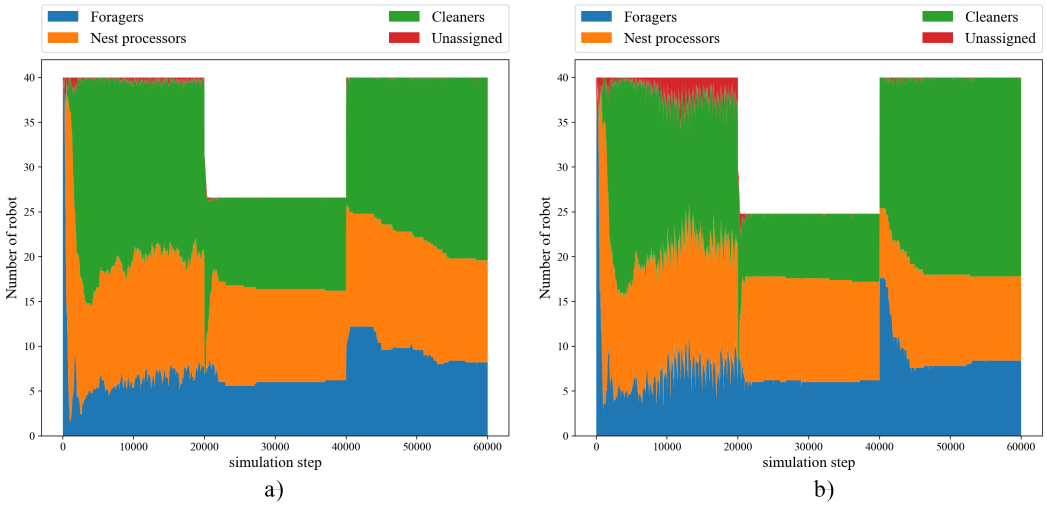


Figure 5.22: Adaptivity in the change of workforce; Labor distribution for a) CAITA, and b) DAITA.

the following observations: It seems that a centralized version of AITA is more suitable if one uses it in an environment where some robots are expected to fail. Conversely, the distributed version is more scalable and should preferably be used when adding new robots is expected. A hypothesis on why this mirrored behavior happens is further elaborated on in the section *6. Discussion and Conclusion*. Overall, both methods offer an effective coverage of lost robots proportional to the demands of the tasks, highly optimizing the task assignment over the entire simulation period.

6. DISCUSSION AND CONCLUSION

Swarm robotics is a fast-growing sector full of exciting literature and new challenges to come. Although the existing body of research partially solves today's swarm robotics' most critical challenges, they usually lack adaptability and improvement opportunities as they are generally centered around one concept. The focus of this thesis has been on providing a software implementation and a thorough investigation of an ant-inspired dynamic task allocation model that claimed to be the foundation of a wide variety of task allocation methods while delivering a near-optimal distribution of labor. The method was based on a clever system of energy supplied and energy demand for a task, information that a worker could access through a binary feedback function. Inspired by the Response-Threshold model, the binary feedback function was an abstracted way to define how agents could sense their immediate environment and tell whether a task was in energy deficit or energy surplus. Because of the binary feedback function, only the required number of robots would be deployed to meet a task's demand, reducing the system's overall energy consumption.

The model has been investigated in a self-developed simulation environment (*3. Simulation Design*) and successfully implemented in a centralized and a distributed way. The centralized version used an area as an information-sharing center where agents reported their current status and were given new task assignments. Oppositely, in the distributed version, the agents shared their knowledge with nearby robots to evaluate the environment's current situation. The environment described a three dependent task system with a foraging task, a nest processing task, and a cleaning task. To highlight AITA's performances and select a bias of noise and communication range that was both challenging and realistic, tests on different levels of communication noise and different communication ranges were performed (*5.1 Experiments on Parameters Settings*). Then, the noise and communication range were used in the experiments to recreate best a physical setup challenged by the laws of physics. The experiments (*5. Experiments and Results*) intended to cover five categories that highlight the strength of a swarm robotics system: Its *robustness*, its *reliability*, its *scalability*, its *versatility*, and its *adaptability*. Moreover, tests over the assumption of the original model's creators, the energy consumption in terms of the number of task switches per robot, and the total covered distance were conducted.

The results of the section *5.1 Experiments on Parameters Settings* have highlighted that both the centralized and the distributed architectures showed strong robustness to

communication failures. However, CAITA revealed weaker performances due to its mode of communication that included a single entity. Communication through a single entity is prone to "single-point-of-failures". If the central tower falls, the entire system cannot communicate, and the distribution of labor is stuck. The effect of single-point-of-failure systems and their relationship to scalability has not been covered in this thesis. However, Jamshidpey and Afsharchi's [10] paper from which the Chapar towers are inspired offers a brilliant analysis and reflection on the problem.

Oppositely, the DAITA system, which included a robot to robot communication system, did not suffer from "single-point-of-failures" as the distributed robots' status and task assignment meant that losing one robot would not limit the communication for the rest of the group.

In addition, DAITA has underlined the weaknesses of using short communication ranges for information sharing since the error E became more significant as the communication range was made shorter. This was a direct consequence of the arena's size, as the robots sometimes had to be separated from a few meters. Nevertheless, the Zigbee wireless communication module envisaged for a potential robotic implementation could be a perfect candidate as it could communicate on longer distances than what was needed for the experiments.

The performance analysis (*5.2.1 - Relationship Between the Number of Robots and the Initial Demand of the Foraging Task*) on the relationship of the number of robots, the initial demand of the foraging task, and the periodical increase have shown that the system's performance was independent of the colony size. Instead, it was dependent on the *system's business*, a metric that described how Θ could be shaped to influence the performance of the system. In order to provide a fair framework of comparisons, initial conditions that best reproduce the *system's business* of the other investigated methods were implemented.

The conducted experiments in section *5.2.2 Task Completion Rate On Different Task Allocation Methods* showed that both DAITA and CAITA achieved a noticeably good task completion rate that was comparable to greedy algorithms and other insect-inspired algorithms such as PSI. Finishing its task first, DAITA performed slightly better than CAITA, underlying a solid relationship between the speed of task completion in relation to the speed at which a system can adapt to the relative demand.

Experiments in section *5.3 Optimization of the Labor Distribution on Different Task Allocation Methods* showed that AITA could reach a near-optimal task assignment regardless of the colony size and the relative workload. Backing up the relationship described above, the fast adaptation to the environment gave an advantage to DAITA, which obtained the highest labor distribution optimization of the two investigated architectures.

The results obtained in section *5.4 Optimization of the Energy Consumption* have

demonstrated AITA's ability to maintain a near-optimal task assignment while maintaining a low energy consumption in relation to the covered distance and the number of task switches per robot. DAITA achieved the lowest energy consumption in terms of the total traveled distance, while CAITA achieved a similar result in terms of the number of task switches per robot. DAITA showed a higher number of task switches per robot than any other investigated methods. This can be explained by DAITA's fast communication system, as a fast information flow provokes considerably more oscillations in the distribution of labor.

Finally, experiments in section *5.5 Adaptive Change in the Workforce* proved that, as the distribution of the current state was done dynamically during the run time, AITA can re-allocate removed robots proportionally to the demands of the tasks, making it adaptive to the change of workforce. It was observed that CAITA offered the best adaptation to the loss of workforce as its optimization of the labor distribution was the best, highlighting great robustness to robot failure. However, it offered the most unfavorable adaptation when the robots were re-introduced (less scalable), highlighting possible improvements in the way information center handles the task assignment. Oppositely, DAITA achieved the worse adaptation to the loss of workforce compared to CAITA and PSI, but the best adaptation to the re-introduction, underlying that the architecture coupled with AITA is highly scalable. This mirrored behavior is challenging to depict, but the explanations might lie in how the information is shared within the system. In CAITA, because only the information center processes and sends out new task assignments, it has a near-perfect knowledge of the current world situation. Thereby, the entire task allocation process is aware of the missing robot's last reported status. Inversely, the information in DAITA is shared robot to robot. Thus, if one robot is gone, only the last few robots it contacted prior to disappearing know about its latest information, thus creating inconsistency in the task allocation process as some robots have more information than others. This explanation might be the answer, though further experiments are needed to confirm this hypothesis.

The implementation and experiments have reflected the complexity of conceptualizing, programming, and solving the near-optimal dynamic task allocation problem for large groups of agents. The self-developed simulation was a critical factor to the findings as programming a lightweight simulation environment that could reproduce the features of real robots and a physical environment has enabled extensive testing and given total control over crucial factors.

The overall results have revealed that DAITA achieves better performances compared to CAITA. This finding means that there is an advantage of using a distributed system when performing tasks involving large groups of robots as the faster communication system displays an excellent swarm adaptation to spontaneous changes in the relative workload of the world.

Although the experiments have not demonstrated a significant performance advantage of using AITA compared to other methods, there is still an important body of improvements. Indeed, an essential block of improvement is the binary feedback function, and more specifically, the energy supplied by a robot to a task. The method implemented for this thesis (*2.2 Implementation: Binary Feedback Function*) was relatively simple and could be improved to obtain higher performance. As AITA achieved similar performances compared to well-studied algorithms, it could be expected to obtain better results through incremental improvements.

This ant-inspired task allocation model has the potential to be used in a wide range of sectors, such as the one mentioned in the introductory part of this thesis. For instance, one could use AITA for victim rescue as its scalability and robustness to communication failures offers a solid alternative to already existing rescue robots. Moreover, as AITA is meant to be a versatile task allocation model offering a large body of possible implementation, it can be adapted to a wide range of tasks.

In conclusion, by implementing a dynamic task allocation method that gives new task assignments based on the current relative need of each task independently of the colony size, by investigating and developing a robust communication system, and by providing measurements of the low energy consumption of the studied mathematical model, this thesis offers a substantial body of work that hopes to solve the main challenges mentioned in the introductory part of this thesis.

6.1 Future Work

As the investigated model has not shown significantly better performances compared to other methods, it could have benefited from other implementations of the binary feedback function. The optimization framework offered by Cornejo et al. is an overwhelming bed of extension possibilities as energy supply based on different variables, such as the robot's characteristics or the current environment, could have been implemented to improve the distribution of labor. Much so that if made unnecessary complex, the task allocation problem could have become NP-Hard, resulting in an unnecessarily complex algorithm.

Nevertheless, improved energy system based on the robot's location can be envisaged. As the robot moves around the arena, the energy it could supply to a task could directly be correlated with its position in the environment. Robots would have a higher chance of switching to tasks related to their current zone rather than being given one following the stimuli strength of the tasks, noticeably reducing the energy consumed by the system.

Consequently, another area of improvement is the robot itself, as adding new sensors to the Thymio-II opens new design opportunities. As discussed by Alexandre Vanini [29], pheromone imitation methods can stimulate the robot to solve a task, i.e., by increasing its

likelihood to switch to a task in relation to what is sensed by the sensors (that is, the more resources from a specific task are sensed, the more likely the robot is to switch to this task). In other words, there is a margin of play in the hardware to improve the energy a robot can provide to a task given its location. In addition, short memory can be implemented in the system. Robots would remember their local environment from a few simulation-step ago and provide more energy if the old environment contained a higher concentration of resources.

As mentioned in the section *5.2.1 Task Completion Rate on Variating Numbers of Robots*, an optimization variable is the location of the different deposit areas within the arena. Indeed, the current implementation makes the robots drop their payload in each zone's central region, which drastically increases the time to find resources as robots will usually be located at the edges of an area due to their stay-in zone behavior. Thus, mechanisms to improve either the placement of the resources within the area or the detection of resources can be implemented. In addition, the congestion problem faced by robots when navigating from an area to another can be lowered by improving the currently implemented robot avoidance behavior. Indeed, the current robot-avoidance mechanism is not optimal and does not offer the best chance for each robot to escape congestion situations. Pitonakova et al. [30] discussed how congestion can affects scalability by limiting communication and information sharing. They proposed a novel congestion-sharing method to solve robot congestion within multi-robot systems. In their method, robots would be sharing information about their current local congestion status to the other swarm members, preventing bigger congestion groups from forming.

The current distributed communication method implemented for DAITA has achieved good results in terms of the investigated metrics. However, the system could improve on information sharing. Even though the system does not contain any single-point-of-failure, losing one robot means its information will not be shared throughout the colony. As soon as some information is kept from other robots for an extended period, it creates inaccuracy in the swarm's current knowledge of the world. Consequently, a global-world-sharing communication system can be envisaged where a robot would share its entire $memoryArray$ rather than its current progress only. The latest status of the missing robot would then be shared within the swarm, making the system more reliable. Moreover, it would improve the performances of shorter communication ranges as robots outside the communication range of other robots could still receive their information (given that at least one other robot is placed in between the coverage areas of the two groups and has received the information).

Finally, as a before-work to the improvement mentioned above, physical implementation and analysis must be conducted to assess AITA's performances in real-life conditions.

BIBLIOGRAPHY

[1] Alejandro Cornejo, Anna Dornhaus, Nancy Lynch, and Radhika Nagpal. Task allocation in ant colonies. In Fabian Kuhn, editor, *Distributed Computing*, pages 46–60. Springer Berlin Heidelberg, (2014).

[2] Mohan Yogeswaran and Ponnambalam S.G. An extensive review of research in swarm robotics. pages 140 – 145, (2010).

[3] Y.U. Cao, A.S. Fukunaga, A.B. Kahng, and F. Meng. Cooperative mobile robotics: antecedents and directions. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, pages 226–234 vol.1, (1995).

[4] Ying Tan and Zhong-yang Zheng. Research advance in swarm robotics. *Defence Technology*, vol.239, (2013).

[5] Pablo García Auñón and Antonio Barrientos. Comparison of heuristic algorithms in discrete search and surveillance tasks using aerial swarms. *Applied Sciences*, vol.8: 711, (2018).

[6] Arne Brutschy, Giovanni Pini, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, vol.28:101–125, (2014).

[7] Levent Bayindir. A review of swarm robotics tasks. *Neurocomputing*, vol.172, (2015).

[8] Khalil Mohamed, Ayman Elshenawy Elsefy, and Hany Harb. A hybrid decentralized coordinated approach for multi-robot exploration task. *The Computer Journal*, vol.62, (2018).

[9] Aleksandar Jevtić, Diego Andina, and Mo Jamshidi. *Distributed Task Allocation in Swarms of Robots*, volume 1, pages 170–193. (2012).

[10] Payam Zahadat and Thomas Schmickl. Division of labor in a swarm of autonomous underwater robots by improved partitioning social inhibition. *Adaptive Behavior*, vol.24, (2016).

[11] Thomas Alves, Jérémy Rivière, Cédric Alaux, Yves Le Conte, Frank Singhoff, Thierry Duval, and Vincent Rodin. *An Interruptible Task Allocation Model: Application to a Honey Bee Colony Simulation*, pages 3–15. (2020).

[12] Frederick Ducatelle, Alexander Förster, Gianni Di Caro, and Luca Maria Gambardella. New task allocation methods for robotic swarms. (2009).

[13] Aryo Jamshidpey and Mohsen Afsharchi. Task allocation in robotic swarms: Explicit communication based approaches. (2015).

[14] Qiuzhen Wang and Xinjun Mao. Dynamic task allocation method of swarm robots based on optimal mass transport theory. *Symmetry*, vol.12, (2020).

[15] Yongming Yang, Changjiu Zhou, and Yantao Tian. Swarm robots task allocation based on response threshold model. In *2009 4th International Conference on Autonomous Robots and Agents*, pages 171–176, (2009).

[16] Corey Binns. Why Ants Rule the World. Accessed: 25 May 2021, (2006). URL `https://www.livescience.com/747-ants-rule-world.html`.

[17] Lori Lach, Catherine Parr, and Kirsti Abbott. *Ant Ecology*. Oxford University Press, (2010).

[18] V.K. Munirajan, Ferat Sahin, and E. Cole. Ant colony optimization based swarms: Implementation for the mine detection application. pages 716 – 721 vol.1, (2021).

[19] Neni Firdausanti and Irhamah. On the comparison of crazy particle swarm optimization and advanced binary ant colony optimization for feature selection on high-dimensional data. *Procedia Computer Science*, vol.161:638–646, (2019).

[20] *ACAI '11: Proceedings of the International Conference on Advances in Computing and Artificial Intelligence*, New York, NY, USA, (2011). Association for Computing Machinery.

[21] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. Fixed response thresholds and the regulation of division of labor in insect societies. *Santa Fe Institute, Working Papers*, vol.60, (1998).

[22] UMSL. Intractable Problems. Accessed: 25 May 2021. URL `https://www.umsl.edu/~siegelj/information_theory/classassignments/Lombardo/04_intractableproblems.html`.

[23] Kevin Leyton-Brown, Holger Hoos, Frank Hutter, and Lin Xu. Understanding the empirical hardness of np-complete problems. *Communications of the ACM*, vol.57: 98–107, (2014).

[24] Thymio | The educational robot to learn, code and create. Accessed: 25 May 2021. URL `https://www.thymio.org/`.

[25] Dieter Fox and Wolfram Burgard. Markov localization for mobile robots in dynamic environments. *J. Artif. Intell. Res.*, vol.11, (1999).

[26] André Carmona Hernandes, Henry Borrero GUerrero, Marcelo Becker, Jean-Stephane Jokeit, and Gregor Schöner. A comparison between reactive potential fields and attractor dynamics. (2014).

[27] A. Alvarez-Aguirre, Henk Nijmeijer, Toshiki Oguchi, and K. Kojima. Remote control of a mobile robot subject to a communication delay. volume 2, pages 55–62, (2010).

[28] Zigbee. Accessed: 25 May 2021, (2021). URL `https://en.wikipedia.org/w/index.php?title=Zigbee&oldid=1025717648`.

[29] Alexandre Vanini. Study of ants' collective behaviours and mechanisms to reflect on possible implementations of software / robotic simulations. (2020).

[30] Lenka Pitonakova, Richard Crowder, and Seth Bullock. Task allocation in foraging robot swarms: the role of information sharing. (2016).