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

Alexandre Jean-Pierre Vanini. [alva@itu.dk](mailto:alva@itu.dk)

Payam Zahadat (90%)

Kasper Støy (10%)

# Abstract – MOK

The paper proposes a software investigation of an ant-inspired mathematical task allocation model created by Alejandro Cornejo et al., which proposes a near-optimal distribution of labor for swarms of robots. [link]. Heavily inspired by the Response threshold model, the algorithm is based on a binary feedback function that senses a worker's surrounding environment and gives information about the different tasks need. By simulating a three-dependent tasks system, the paper reflects on the model's ability to reach equilibrium in the distribution of labor while maintaining low energy consumption cost in terms of the number of task switches per robot, the covered distance, and the task completion rate. Moreover, an analysis of the system's robustness by changing the communication range and adding noise to the system is conducted. The algorithm is successfully implemented in two architecture of the swarm robotic paradigm; A centralized version where a communication tower is used to transfer the information robot to robot, and a distributed version where the information is rather shared from a robot to another. The conducted experiments show that the algorithm can reach near-optimal labor distribution as a function of the colony size and the current task needs.

It aint supposed to be in function of the colony size.

# Introduction - OK

Swarm robotic 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 » [1]. Moreover, it is "a property of systems of non-intelligent robots exhibiting collectively intelligent behavior" [2].

The studies of task allocation systems related to swarms of robots are mainly used to understand how a complex system of tasks can be solved using multiple agents [link 2-4 symmetry]. The tasks can either be achieved by an individual, alone or by a group of robots, cooperatively [brutshy]. Distribution of labor in groups of individuals can be considered done in two ways: Given that the environment is limited in its growth, one can implement a pre-determined task allocation strategy, such as robot-planning [LINK], where the individuals can perform a set of given tasks but are unable to adapt to dynamic change in the environment. Conversely, the focus of this paper is turned towards task allocation in dynamic environments where autonomous robots are given behavior-based mechanisms and adaptive task allocation methods to respond to unplanned events spontaneously. The main challenges faced by swarm robotic when dealing with dynamic environments are the following:

* 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 strategy 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
* Scalability and task allocation dynamicity is hard to achieve through limited strategies

Alejandro Cornejo et al. [link] propose a mathematical model for dynamic task allocation inspired by ants and their collective social behaviors in relation to the distribution of labor. Their work gives a framework to study different implementations of task allocation by providing a set of helpers and functions to frame the problem. In this work, the author's contribution is to provide a software implementation of the beforementioned mathematical model to resolve the abovementioned challenges 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

The investigation of the provided ant-inspired mathematical model is divided into two architectures of the swarm robotic paradigm; Firstly, a centralized architecture (see figure n), 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 [link]. 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 [link]. Moreover, this 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 [link]. The second investigation is in regards to the distributed architecture. This time, the information is shared among all individuals through local communication. Each robot shares its state and is responsible for understanding their environment and assigning a task to themselves. 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 does affect one robot and not the entire communication process.

In order to evaluate the mathematical framework with its two architectures, an environment simulation is defined, and experiments are conducted. The first set of experiments observes the effect of communication disturbance with respect to different levels of noise and different communication range 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.

[link file:///Users/freak/Downloads/bxy107.pdf]

The organization of the paper is the following. Section 2 describes related works. Section 3 introduces the ant-inspired algorithm. Section 4 introduces the methods and framework used in the system. Section 5 introduces the experiment while providing results and discussions. Section 6 discuss and concludes the work. Section 7 provides a framework for future work.

# Related work - OK

Models that try to solve dynamic task allocation have already been experimented with, such as the work of **Aleksandar Jevtić et al. [?]** where the authors create 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 plays a role in the task allocation distribution. The model has shown weaknesses in the distribution error under specific parameters settings where deployment cost was decreased, resulting in a higher distribution error. However, the experiments conducted on the algorithm have shown that their algorithm offers excellent scalability in terms of the number of robots and the number of tasks.

**Ducatelle et al. [?]**  propose two methods for decentralized concurrent task allocation for groups of flying robots in a confined area where novel communication methods are used for share task demands globally. The first method is based on interaction through light signals-based communication, where robots react to different light signals and colors dependent on the current task distribution. The second is a gossip-based method where the infrared sensors present on each robot allow primitive communication of task status and robot status. The results concluded that gossip-based communication using infra-red sensors was more efficient than using other sensors to detect emissions of light by other individuals in a limited environment. However, they also concluded that the gossip-based communication was subject to packet loss and, therefore, is less scalable.

**A. Jamshidpey and M. Afsharchi** [?] propose a study of multi-robot cooperative task allocation in a dynamic, unknown environment through different communication-based approaches. They define the following four communication-based approaches: The first one is a *static communication-based method* where a robot’s task allocation remains unchanged until it runs out of *search energy*. The second one is an improved version of the first method called the*dynamic communication-based approach,*in which when a robot runs out of energy, it can choose its next task following a probabilistic model. The remaining methods (a decentralized and a centralized one) use Chapar towers (a radio communication station inspiration) placed in the environment and act as an information center. In the Decentralized version, groups of robots called the “Chapars” transmit information from the Chapar towers to the workers and so forth. In the centralized version, one Chapar station covers the area, and the robots report to it. The conclusion drawn is that there is no significantly higher efficiency when comparing the centralized or decentralized version, but single points of failure and robustness issues in communication are highlighted.

**Jérémy Rivière et al**. [?] propose methods of self-awareness in robot task selection based on the *Response Threshold Model* (LINK TODO), where individuals can decide whether to pursue a task or to stop it independently of the task stimuli. The results have shown that the swarm can reach equilibrium.

**Qiuzhen Wang and Xinjun Mao 1 [symettri?]**propose an algorithm based on the Optimal Mass Transport theory (OMT) designed to solve the problem of large-scale tasks. The experimental results show that there exist efficient solutions to large-scale dynamic environments within the distributed paradigm.

**Zahadat et al. [?]** propose a honeybees-inspired algorithm call the Partitioning Social inhibition Method (PSI) and demonstrate that through local interactions, the method can achieve task equilibrium regardless of the colony size and the relative workload.

**Yongming Yang et al. [?]**propose a distributed algorithm based on the *Response Threshold Model* that does not utilize any communication medium. This research has shown that such an algorithm can successfully adapt to the demand and task completion growth via a solid labor distribution regardless of explicit communication within the swarm individuals.

**Finally, Arne Brutshy et al.** [?] propose another communication-less task allocation method based on interaction rate within individuals using a set of interdependent tasks and prove that the method can reach near-optimal task allocation.

# Ant-Inspired Task Allocation Algorithm

This chapter is an overview of the AITA mathematical model. Please refer to the original Alejandro Cornejo et al’s research paper for a more in-depth presentation of the algorithm.

Introduction

Ants are incredible social insects in many ways. From their adaptability to a great range of environments to the extremely high level of self-organization in colonies, sometimes reaching up to 10 million individuals [link], they are a great source of inspiration for robotic and swarm intelligence. Their most exciting feature is their maximization of the distribution of labor to sustain the colony’s need to periods of over decades and how it can be applied to today’s swarm robotic challenges. Alejandro Cornejo et al. [link] have designed a mathematical framework inspired by ant’s collective behaviors, which at the time this paper is written (Q2- 2021), it has not been implemented nor tested in a simulation environment. This project attempts to give this algorithm a fair implementation and experimental comparisons 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 » [TR link in note]. However, in this implementation, the individuals cannot quantify the stimuli but can tell whether a task needs more attention or not.

# Mathematical framework

Alejandro Cornejo et al. introduce four quantifier helpers that help a worker telling whether a task is in energy deficit or surplus. To begin with, they define the function **d**(T, t) as 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, and many more. Secondly, the function **e**(T, a, t) yields how much **energy** [NDPDB] an ant can provide to a task t at a given time T. The **energy**an ant can provide to a task depends on environmental variables, ants’ characteristics, and previous experiences of the specific task. Thirdly, they define the **energy supplied** to a task t at a given time T **w**(T,t) as the sum of the **energy** **e**(T, a,t) currently provided by all ants performing a task. Finally, they define **q**(T,t) as being **d**(T,t) – **w**(T,t) (referred in this article as the “**energy difference**”), or; the current demand for a task minus the energy currently supplied by all ant to that task at a specific given time T.

TODO NDBDP:

*As described in the original, the energy unit can be any kind of energy (watt, joules, etc.) as long as one keeps it consistent throughout the implementation. This paper does not attempt to quantify or to the label of much is one energy.*

**Assumptions**

Alejendro et al. (nametodo) define a satisfying task assignment as one where no task is in energy deficit. In other words, they seek a task assignment where the set of task reach equilibrium – I.e., where the **energy supplied**to a taskmatches the task’s **demand.** To evaluate the performances, they define an optimal task assignment as one that minimizes the square difference between the energy demands and the energy supplied. Furthermore, they assume that if one task can be set to an equilibrium, all other tasks will be served.

**Model restrictions -> does this section belongs here?**

The section “Mathematical framework” mentions four helpers. However, it is still unclear how they exactly work as it only has been mentions of potential “environmental variable” for the demand or “ant’s characteristics and experiences” for the energy. The environmental variables, ant’s characteristics, and experience settings are so vast that it is impossible to include all of them in the model. Indeed, even Alejandro Cornejo et al. have decided to leave this choice to someone else who would implement the task allocation they have created, as they highlight how the complexity of individual variation quickly results in an “intractable task allocation formulation.” Being an intractable formulation means that there is no efficient way to solve the task allocation problem [Link]. Intractable problems are commonly referred to as NP-complete problems [Link].

As will be elaborated later, the simulated individuals are a set of homogeneous robots 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, since the simulated environment is controlled and not meant to reflect challenging environments such as a jungle or a desert, but instead meant to reflect the use of such a system in places such as a depot or a hangar, its effect is limited. Finally, it has been chosen 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. What it means for the implementation is that one robot can carry out a single sub-task at a time, and once this sub-task is completed, the demand in the parent-task decreases by one (the task are furthermore discussed in the section “TODO”). Implementations of systems where the energy an individual can supply to a task depends on its characteristics and other variables are further discussed in the section “future work”.

**The binary feedback function**

The algorithm is based on the worker’s ability to sense its environment through a binary-feedback function f(T, i). Recall the helper function q(T,t), which is the energy difference for a task t at a given time T. The binary feedback function yields 1 if the energy difference for a task is in equilibrium or energy surplus, -1 otherwise (note that the binary feedback function does not provide enough information for the workers to tell whether a task has reached exact equilibrium). The paper also introduces other binary feedback functions in their further work section, which will not be covered in this paper. Sensing the energy difference for a task through a binary function means that a worker cannot quantify 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.

**Algorithm**

The algorithm (which considers the size of the colony |A| as being fixed) works as follows: Firstly, all worker executing the algorithm maintains a current task *currentT* and can be found in one of the five following states: Resting, FirstReserve, SecondReserve, TempWorker, and CoreWorker. Moreover, workers carry a table of potentials Q for each task used to determine which task a worker 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 candidate list that contains tasks with a potential of 3. With equal distribution, a worker will choose a task from the candidate list and leave the Resting state 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 currentT ask (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 currentT ask, 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 currentT ask, 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 currentT ask, and otherwise transition to the TempWorker state.”*

For more detail, refer to the original paper.

The complete original algorithm is shown in Figure N. Figure B depicts a correction made to the algorithm, explained in the section “correction on the original algorithm TODO.”l

**Original algorithm correction**

Throughout the implementation and tests conducted on the original algorithm, it has been found that there was an issue with the increase of potential for a task in table Q (see Figure N, todo line x to y). 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 it’s energy potential is 3. However, it is not given that the task is selected. Indeed, workers select a new task with probability ½, meaning that first, a task of potential equal to 3 can re-enter the code on todo line x to y, and end up with a value higher than 3, which defeats [todo paper name] ’s definition of the value the potential for a task can take in the Q table:” [..] The potential for every task is a two-bit value {0, 1, 2, 3} [..]”, and secondly if all tasks are higher than 3, none of them are added to the candidate list, leading the robot to be stuck without any assignment until eventually, a task is in energy surplus its value goes back to 0 in the worker’s table Q.

To overcome this situation, line 3 in figure N has been updated to

…

And line N todo has been updated to:

..

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

Table

Description automatically generatedA picture containing table

Description automatically generated

**Here, redo the algorithm as I know how it should be, no copy past the algo from pseudo code**

## Integration

This section describes how the AITA algorithm is integrated into simulated agents and what design choices have been implemented. As the before-defined algorithm is only a brick of the global implementation, it is essential to explain how the functions of the individuals are made and how they shape the performances of the algorithm. Among others, the feedback function and how it is integrated is discussed. Moreover, this section covers the communication system for CAITA and DAITA.

### Binary Feedback function

The binary feedback function proposed by AleTODO 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. A near-perfect implementation of the binary feedback function is crucial to obtain the best results—the 2. Related Work section is an excellent source of inspiration to design this 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. As the world is limited in its global information, direct access to data on task needs is hidden from the swarm. The sub-goal is then to create a communication medium to share throughout the swarm the knowledge of each robot on its current progress on each task. The first method proposed by [TODO] is 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 stimuli 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 are more prone to noise and faulty behavior [sourcetodo?].

Conversely, one can choose to implement radio-telecommunication through transmission tower such as in ???work where radio towers are placed on top of the agents to cover a wider communication range than basic built-in IR-sensors. This implementation works well and allows integration on both a centralized version of AITA, where each agent communicates with a radio-transmission tower given that it is in the coverage area, and on the distributed version where each robot can be equipped with such radio-transmission tower and communicate with one another. This paper 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.

**Centralized and Distributed**

As mentioned in 1. Introduction, the beforementioned algorithm has been implemented in both a centralized fashion - where a communication area (the nest) serves as an information center - and in a distributed fashion where the information about the demand and the current task assignment is shared globally throughout all the individuals.

**Centralized AITA**

The centralized ant-inspired task allocation implementation or CAITA works as shown in figure N; At initialization, starting demands are assigned to the information center by an outside person. As the simulation goes, robots have to periodically drive back to the information center area and share their current status [NDBDP]. 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 these pieces of information, it updates a table of global knowledge which contains information about every other robot, decides on and sends back a new task assignment for the robot. In the CAITA implementation, the binary feedback function is used by the transmission tower / information center (unlike DAITA where each robot uses the feedback function).

Mechanisms for improving the robustness of the system against potential robots’ failures are implemented. Each time a robot does not report within 2000 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 place for another potential idle, first reserve, or second reserve robot to take over its work.

NDBDP: (Note that: a robot is not assigned a new task if it carries a payload).

**Distributed AITA**

Significant changes are made in the distributed version of AITA (see figure TODO). 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 tasks, 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, these pieces of information are then 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. Before reaching the end of its simulation step cycle, a robot runs the AITA algorithm with its current knowledge of the world situation and is attributed a new task or not depending on it.

Robustness mechanisms similar to the one implemented for CAITA are also present in DAITA. This time, each robot keeps track of the last time a robot has contacted it. Suppose this elapsed time goes over a specific pre-defined time. In that case, the robot is considered gone from the other robot’s system. The total energy applied to a task decreases, leaving place for another potential idle, first reserve, or second reserve robot to take over its work.

**RNDTA, GTA, and PSI**

There exist many methods of algorithm evaluation. One of them is to search for and implement an algorithm that solves the same tasks as CAITA and DAITA and that relevant for comparisons. This section provides an overview of RNDTA, GTA, and PSI; three algorithms choose to run against AITA. These algorithms are implemented in the same evolution environment as AITA and go through the same experiments.

**Random Task Allocation Algorithm**

The random task allocation algorithm, or RNDTA, is a method where each individual, is attributed a new task every 600 simulation step [NDBDPTODO] (after thorough experiments, 600 simulation step has been seen to be the most optimal task time) following a uniform distribution (that is, no task as more chance to be selected over another). Since the robots do not require sharing any kind of information and do not need to be aware of the current world state, this task allocation system does not suffer any kind of communication failure or overhead, making it highly scalable and robust.

This algorithm is choose as it is the simplest algorithm one can design. Thus, it serves as a lower boundary as to what the ant-inspired algorithm (and any other elaborated algorithm) should not go below. It is expected that this algorithm performs the poorest as it does not take into account the current world’s state and is very inconsistent.

**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 prone to suffer the same challenges; 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 a high computational power consumption.

### 

**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. [link]). 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.” Its adaptivity in terms of the swarm size and the relative demands is an essential aspect of modern task allocation. 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 environmental setup conceived for this project describes how the specific variables of PSI are adapted to this paper’s system.

For this research paper, Payam Zahadat has been kind enough to provide the student with the C++ code with which the simulation was run. The algorithm has then been transferred to the Python code the author is working with and adapted to the current environment and communication mechanisms. This adaptation means that PSI executes the same set of tasks as all of the 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 the consequences of applying the algorithm to the author’s environment.

**Algorithm**

Each robot of the swarm holds an x value that represents their physiological age. This value represents 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 of *xmin* and *xmax*, where the range is split equally by the number of tasks so that each task gets the same amount of distribution (see figure n todo). 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 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 interaction with the swarm members, but this paper does not intend to cover that. Please refer to the original paper for further information on how the local interaction within individuals changes the x values.

As mentioned above, PSI has not been designed to run under the same environment 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 project. The two impacted areas are the demand and the specific condition under which a worker can be allocated a new task.

**Demand**

The current implementation of the environment yields that the demand for a task can vary from inf to inf. In contrast, in Zahadat et al.'s system, the tasks have a value representing a fraction of the current demand and usually variates 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. The system designed to adapt PSI's demand to AITA's environment is the following: The demand is mapped to a 1 - 20 scale.

Imagine the following sequence of demand; [23, 132, 12], where 23 is the current demand for the foraging task, 132 is the current demand for the nest processing task, and 12 is the demand for the current cleaning task.

**TODO** put that into equations:

# First, the highest demand is determined

max\_demand = max(23, 132, 12)

# Then each demand is mapped to the 1-20 system following this equation where x is the input demand:

f(x) = math.ceil(x / max\_demand \* 20)

return f(x) if demand > 1

1 otherwise

**The condition under which a worker can be allocated a new task**

As for all the other algorithms, a PSI robot cannot be attributed to a new task if it is currently performing one. It means for PSI that the value x of each individual is delayed as long as the worker is currently carrying a payload.

## PSI starting variables (TODO MOVE TO PSI DEF?)

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

A wide range of tests and experiments conducted on the adaptation of PSI's demand and task attributed and the different variables 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 get as close as possible to the performances of PSI under its original environmental setting.

TODO add table.

|  |  |
| --- | --- |
| Xmin | 0 |
| Xmax | 512 |
| Noise on x | 0 |
| Phi | 1 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

In order to evaluate the performances of the proposed methods, different metrics are taken into consideration; The task completion rate, the total covered distance for all robots, the distribution of robots over the different tasks, the number of task switch 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”:

The swarm’s perception error also describes the speed at which the information of the demands of the different tasks is shared throughout the swarm.

Methods

The experiments intend to assess the efficiency of AITA over five categories that are commonly referred to as being what a swarm robotic 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. Finally, the system’s *reliability* states that the robot should be consistent in its probability of solving a given task. Along with these five categories, the systems are tested upon Alejendro TODO’s metric of stable task allocation system – the minimalization of the squared difference between the energy demands and the energy supplied. Moreover, an in-depth performance analysis of AITA’s energy consumption in terms of the task completion rate, the covered distance by all robots, and the number of task switches is conducted.

TODO : Each experiment is run five times and averaged to give a more fair outcome than a single run.

# Experiments

## Environment framework

The different task allocation methods, CAITA, DAITA, PSI, RNDTA, and GTA, are tested and experimented on in an agent-based simulation built by Alexandre Vanini and written in Python [LINK TO GIT]. The model of the simulation (figure N) 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 (blue, 1.4 meters x 1.4 meters), where resources collected from the outside world are stored. Moreover, the dump area is also used as the information center (where robots report their status) when running CAITA. The transit area (pink, 1.4 meters x 1.4 meters), where resources processed from the dump area are stored. Finally, the waste area (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 (the 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). The topology of the world is a rectangle box bounded in all its directions which no agent can traverse.

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 and collect resources to then bring them back to the dump area. Workers carrying out the foraging task are shown in red. Thirdly, nest processors – i.e., process 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. 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 *CleaningTask* will only be able to sense resources of type *waste*.

This set of tasks (Idle is omitted as it is more a state than a task) are three dependent tasks that mean 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 has first to be collected in the foraging area and brought back home, and for the demand to rise in the cleaning task, a resource first has to be collected outside and brought back home and then has to be processed by the nest processors and move to the transit area. This three dependent task setup 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 set of tasks 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.

Whiteboard

Description automatically generated with medium confidence

## Environment setting

The current simulation environment yields the following assumptions and variables; Firstly, it assumes that all the simulated agents are a set of homogeneous Thymio-II robots [link?] (a pre-built ready-to-use robot equipped with built-in sensors and IR communication system). 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 somewhat precisely move 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. Moreover, 2000 resource items are distributed within the *foragingArea*. The robots do not have any degrading variables and remain as performant as they start throughout the simulation.

For the CAITA system, 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. Moreover, 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. In order to simulate this behavior, a random transmission receiver mechanism is implemented and randomly decides from which transmission a robot will receive its packet. The distribution is uniform; each packet is given the same chance to be received. This system ensures fairness and close-to-reality communication transmission. GTA and PSI, who use the DAITA communication system, suffer the same communication restrictions and challenges.

### Mutable variables

Along with assumptions, a set of specific 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 performances and results depending on its value:

- The number of robots

- The noise/probability of communication failure

- The communication range

- The demand for each task

- Periodical Increase of the demand in the foraging task

A table with each variable’s value can be seen at TABLE N TODO

### Increase in the ForagingTask’s demand

The periodical increase in the *foragingTask* 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 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 achieve a task. 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 a busy as it can get. 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 changing.

Tests with 40 robots have proven that increasing the demand by seven resources at every 500 simulation step is suitable to achieve system business. If one chooses to use an increase of 6 or less for this experiment, it will keep the system in a lazy state, with more or less half of the workers busy on average (dependent on the increase), over the whole period.

***------ IN OVERLEAF ABOVE***

***------ IN OVERLEAF ABOVE***

***------ IN OVERLEAF ABOVE***

**Experiments on parameters settings**

In order to best see the robustness and reliability of the CAITA and DAITA system 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.

**Noise**

The noise is implemented as a communication failure mechanism. That is, whenever a robot tries to broadcast its current knowledge of the world (DAITA) or report to the information center (CAITA), there’s 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, and Pnoise is gradually increased to reach Pnoise = 0.99 finally.

**Noise DAITA**

As shown in figure N (todo), the different levels of tested noise don’t show any significant variation in the task completion rate (the rate at which the given task is completed). Nonetheless, a slight variation of the completion rate for Pnoise = 0.99 can be seen in Figure N, where the system completes its task sooner than the others.

Chart, line chart

Description automatically generated

These variations could mean that the system is highly robust even when 99% of the communications are lost. However, to furthermore explore the incidence of the noise on the system, one can look at swarm’s perception error e of the actual environment.

Figure N depicts the metric e for Pnoise= 0, 0.3, 0.7, 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 averaged error of 1.40 resources, 1.42 resources, and 1.47 resources, for Pnoise = 0, Pnoise = 0.3, and Pnoise = 0.7 respectively. These slight variations, even at high noise levels such as Pnoise = 0.7, 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 each simulation time 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 shared information globally. However, when the probability of success drops at 0.01, the robot is receiving on average 0.39 packets each simulation step, or less than one each round, which means that successful communication can take up to on average 2.56 rounds to happen. Delayed communication can be seen in the swarm's perception error e of the actual environment shown in figure N were for Pnoise = 0, the tendency skyrockets to a peak of 25 resources. Nevertheless, given enough time, the error e plummets, and the system stabilizes to reach just above the same error e as the other, or 5.4 resources on average. This adaptation means that under the before-defined environment and these specific starting conditions, the system is robust to communication failure over time.

As the probability of communication failure reaches a high level, the success rate 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 M where when 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 (see figure n), 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 switch 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 N where, as the demand for resource collecting gently drops, the average sensed demand (***in yellow***) is slightly shifted compared to the actual demand (***in green***). 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 (collecting more resources than needed by the *foragingTask*). Over-collection of resources explains why the curve slightly variates for Pnoise = 0.99 in figure N and why the system finishes first. Indeed, over-collecting compensates for the periodical increase in the *foragingTask*, which over short periods gives a time advantage to [NDBDP]. However, faulty and inconsistent distribution of labor proves that high levels of noise are harmful to the system.

NDBDP: It gives a time advantage under the specific set of assumptions, tasks, and environments defined for this paper.

Chart, histogram

Description automatically generatedChart, line chart

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated |
| Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated |

TODO add foraging demand graphs for each

Maybe in legeng of row[3], the no foraging at 2000 can be seen in the error graph (it is the spike). tODO above, can’t tell which graph is what lol

**Noise CAITA**

**TODO verifier si c’est bien du 50:7**

In the case of CAITA, the task completion rate shown in Figure N demonstrates signs of weakness when the level of noise Pnoise reaches 0.99. However, the other levels of noise achieve the same performances. Recall that the communication system of CAITA is fundamentally different from the one implemented on DAITA. In CAITA, each worker has to report to a central, which then distributes tasks. Back and forth communication means that there is twice as much chance that a communication between a transmitter and a receiver fails, worsening high levels of noise effects.

Chart, line chart

Description automatically generated

The communication failure effects can be observed in Figure N (swarm perception e for Pnoise = 0, 0.3, 0.7, 0.99). For a noise level of 0.99, the error e gets worse over time, reaching an averaged error e of 19.32 resources. Oppositely, for levels of noise Pnoise = 0, 0.3, and 0.7, the averaged error e lowers over time, reaching 6.84 resources, 6.82 resources, and 7.44 resources, respectively. The probability of receiving a packet at a given round for Pnoise = 0.3 is of 0.3 \* 0.3 \* 39 = 3.51, and of 0.01 \* 0.01 \* 39, or 0.0039 for Pnoise = 0.99.

Chart, line chart

Description automatically generated

In conclusion, both DAITA and CAITA have shown a high level of robustness to communication failures for noise levels reaching up to 70%. However, both systems have demonstrated 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 failures. Moreover, it does not show any recovery over an extended period (see figure N todo). Oppositely, DAITA shows signs of recovery as its trend peaks down (see Figure todo N) 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 level of noise, 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 high level of noise are not covered in this paper, but the section ??? in the work of ???? [TODO] provides a good understanding of the problem.

**Chosen Pnoise parameter**

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

### Communication range

As the system uses radio-telecommunication devices to communicate on the current status of the world and to report their status (see section integration TODO), experiments on effect of different length of range are relevant to this project. The experiments assumes that a communication device such as …. [TODOlink] which can communicate up to ???, 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 (CAITA has no range, rather it assumes a worker can communicate with the information center as soon as it enters the communication area), which task is to collect and fully process 150 resources with 40 robots. The range is first of 0.1 meter and increases to reach 13 meters, which covers the entire arena. The task completion rate results for range 0.1m, 0.5m, 1m, 5m, 7m, and 13m are shown in Figure N.

As this part is experimenting on noise range effects, noise in communication is set to 0.

Chart, line chart

Description automatically generated

As visible in Figure N, 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 offer the most consistent result and a high task completion rate, with a very similar trend. Finally, communication ranges of 0.5 and 1 meter’s trends show slight variation as they complete their assignment faster than any other communication ranges. As for the section NOISETODO, the task completion rate is not enough to demonstrate the effect of shorter communication ranges over the AITA system. The swarm’s perception error e of the actual environment is recorded and shown in Figure N to highlight possible issues with shorter ranges.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Figure N shows that communication ranges of 0.1m, 0.5, and 1 m offer the worst speed in global information sharing, reaching an average error of 304.69 resources, 29 resources, and 10.62 resources, respectively. Conversely, communication ranges over 1 TODO chiffre ici more meter offer a low level of error, reaching an average error of 1.45 resources overall. **Although the effects of short communication ranges are consequent on the system, recovery can be observed as short-communication ranges' trend go down and stabilizes over time (see curves for ranges 0.1, 0.5, and 1 meter in Figure TODO).**Nonetheless, harmful effects due to short communication ranges can be seen in the number of task switches per robot shown in Figure N (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, over systems with long-range communication, such a 5 or 13, the global information is shared faster, resulting in a higher number of task switches per robot.

Short-range communication distance's effects can be correlated to the *knowledge error shift*discussed in section NOISETODO. 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 series of graphs in Figure TODO show the real need and the averaged sensed need by all robots in the environment for ranges of 0.1, 1, 7, 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 the simulation goes on, resulting in over-collection. As the communication range gets longer (graphs for 1, 7, and 13 meters), the shift gets less significant, and the over-collection effect fades out.

A solid feature of the environment directly related to a healthy system is the visible periodical peaks in the distribution of labor as seen in the series of graphs in Figure N. These periodical peaks happen as the demand periodically increases, a sign 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 0.5 meters, respectively. These two graphs do not show peaks. Inversely, the two other graphs (distribution of labor for range 7 meters and the entire arena) show signs of swarm adaptation as the periodical peaks are visible throughout the simulation period.

Meh, to rewrite.

|  |  |  |  |
| --- | --- | --- | --- |
| Chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated | Chart, histogram  Description automatically generated |
| Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated |
| Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated |

0.1 TODO inverser les graphs ligne 2 et 3. 1 7 13

**choose communication range parameter**

## For the rest of the experiments, the communication range is set to cover the entire arena, 13 meters. Although communication ranges of 5 and 7 meters offer high consistency while reducing the swarm’s perception error, it also underlines that the higher the communication range gets, the less significant the improvements are. Thus, and for the sake of simplicity, 13 meters is chosen as it will offer the highest consistency and accuracy in the results of the experiments.

## Experiments

Table N TODO shows the set values for the variables for each experiments.

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Describes |
|  | 2000 | Number of resources |
|  | 13 meters | Communication range |
|  | 0.3 | Probability of communication failure |
|  | 500 simulation step | Period at which increase *ForagingTask* demand |
|  | 0 | *NestProcessingTask* initial demand |
|  | 0 | *CleaningTask initial demand* |
| TETA | See experiment settings | The relationship between the Number of robot – The initial demand of the *foragingTask* – and the periodical increase of the *foragingTask* demand |

*TODO, try to have something more scientific and call all :ForagingTask, NestProcessingTask, CleaningTask for instance.*

## 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 the task, the better the system is. In order to experiment with the task completion rate, experiment 1 is defined. Experiment 1 consists of collecting, processing, and cleaning 150 resources as fast as possible [NDPDB]. TETA is set to (R:50:7) where R is dependent on the section.

**Task completion rate on variating numbers of robots over CAITA and DAITA**

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 N and N, 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 struggle when this number goes above 50. Using 70 robots or 100 robots is barely as good as using 50 robots to achieve the same performances in the former. 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. Many variables can play a role in the drop of performances observed when reaching higher numbers of robots. The first variable is robot congestion (as shown in the figure) – 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 de-congestion algorithm are possible, however not covered in the experiments (although reflected in section TODO). 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 performances, in an environment of 70 or 100 robots, AITA will only deploy as many robots as needed to cover all tasks. It leads to a significant part of the colony remaining idle (creating more congestion) if the demand is not as high as needed to occupy 70 or 100 robots. This behavior can be considered “underperformance”, but it is just the AITA algorithm not using more energy than needed to complete a given task. The relationship between the number of robots performing the tasks and the initial demand for the *ForagingTask*is elaborated in the next section TODONAME.

[NDBDP] : Experiences have been stop at 40’000 simulation as some experiment could take over 200’000 simulation steps

?.Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

“Figure N: Mild congestion of the system. As more robots join the same task, more of them need to gather to the same area, generating a traffic jam in the nearby surroundings.”

A picture containing scatter chart

Description automatically generated

**Relationship between the number of robots and the initial demand for the *ForagingTask***

The relationship between the number of robots and the initial demand of the *ForagingTask*can be written as follows: Nb*robots*:Init*foragingTask, or TETA*. This relationship is tightly bonded to the fixed periodical increase of the *ForagingTask* demand discussed in the section (TODO NAME OF SECTION). The relation can then be rewritten as follows: TETATODO = <Number of Robot:Initial Demand ForagingTask: Periodical Increase ForaginTask Demand> *Periodical Increase ForaginTask Demand*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 TETATODO where the workload of the environment is low, keeping a large population of individuals idle (see Figure N ). Inversely, Figure A shows a system where TETA (give teta number) is configured to obtain a high workload, keeping the system busy over the simulation (TODO).

To further observe the effect of different TETA values on the optimization of the task completion rate, the series of Figure N to A shows the results of experiment 1 (using DAITA) conducted with the following TETA parameters: <100:25-50-75-100-150-200:7>.

Chart, line chart

Description automatically generatedTODO figure N: Labor distribution over a simulation. The red part are idle robots.

As can be observed, the choice of different values for TETA highly influences the result obtained. The trends show an optimization of the task completion rate as the number of robots grows. However, reached a tipping point of the initial demand set to 100, the performances are weakened as it takes more time to collect 150 resources since the workers' force is focused on the ForagingTask.

**Task completion rate on different task allocation systems**

The same experiment is conducted on RNDTA, GTA, and PSI with a parameter TETA set to (40:50:7) and compared to the two previous systems. In Figure TODO, 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, achieving the worse task completion rate (100% at 12536 simulation step).

The AITA system can then be seen as performant and fast to complete its set of tasks compared to other systems, although the centralized version is expected to perform slightly worse.

Chart, line chart

Description automatically generated

## Robustness and Energy consumption

ALENJENDOTODO 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 is. To prove that their system can reach near-optimal distribution of labor, Experiment 2 is defined. Experiment 2 (TETA is set to <40:50:7>) is a long-run simulation of 30’000 simulation steps (no goal) to demonstrate how AITA minimizes the inconsistency in its task assignment and reaches near-optimal labor distribution.

**Near-optimal labor distribution proof on different task allocation methods**

Figure N shows the squared difference 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 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 (Recall over-collection from section TODO). As it over-collects, the demand in each task is not met, and the growth never stops. Oppositely, PSI, with an average squared difference of 99, CAITA, with an averaged squared difference of 96, and DAITA, with an averaged squared difference of 44, show great performances as their trend only diminish over time, reaching near-optimal task allocation. GTA shows the same downward trend as DAITA, CAITA, and PSI. However, its averaged squared difference over the same period is 186, which underlines high inconsistency in the distribution of labor, far from reaching a near-optimal curve.

Chart

Description automatically generated

Legend: As it is significantly performing worse than the other TA’s , RNDTA is shown in a transparent gray and its scale is on the right rather than on the left.

TODO say that CAITA is a bit worse

Both CAITA and DAITA offer the lowest squared error time, eventually reaching an almost null value as the distribution of labor’s performances are so high that the task’s demands are all covered. This comparison proves that over time, AITA reaches near-optimal labor distribution.

# Energy consumption title

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

**A Comparison of the Total Distance Traveled by Robots with Different Task Allocation Methods**

The total covered distance of the entire swarm reflects the energy consumption of the different task allocation methods. The more distance is traveled, the more energy a system has consumed over its lifetime. The relationship between energy consumption and covered distance is an optimization goal the different methods have to optimize.

Chart, line chart

Description automatically generated

Figure N shows that over a given period, DAITA uses less energy by having covered a distance of 12816m, 557m less than the second to best in energy consumption, PSI, with a covered distance of 13373m. PSI, GTA, and CAITA cover almost the same distance with 13373m, 13451m, and 13591m each. In comparison, RNDTA has a high energy consumption with a total covered distance of 14801m. The performances achieved by DAITA can be explained as the AITA algorithm has been designed to optimize the number of allocated workers to a task related to 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 less energy. Unlike DAITA, workers in CAITA have to report their status to the information center periodically. It means that there is a likelihood that a task’s energy demand is met while another robot is unaware of it, effectively using more energy as the robot could have had stopped moving if aware of it.

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

**A Comparison of the number of task switch per robot with Different Task Allocation Methods**

By minimizing the number of task switches per robot, task allocation methods reduce the system's disturbance, resulting in less energy used over the simulation period. Figure N 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 task switches per robot [NDBDP]. CAITA is the one that minimizes the optimization variables the best, reaching on average 29 task switches per robot. RNDTA and GTA obtain better results than DAITA (with 56 task switches per robot on average), using 29 and 44 task switches per robot on average. Comparison is hard to make between DAITA, GTA, and RNDTA as both GTA and RNDTA are deterministic algorithms. This time, the periodical report has given an advantage to CAITA as workers ask for a new task assignment less often than for DAITA. Moreover, CAITA's curve is less volatile than DAITA's, meaning that its distribution balance is better.

[NDBDP]: Recall that PSI suffers from the difference between its original environment and the one used for this paper.

Chart, line chart

Description automatically generated

TODO LEGEN SAY PSI IS ON THE RIGHT

TODO Also explained that minimizing task switch is minimizing energy consumption as robots stay consistently in the same task

## **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 (see section TODO discussion). Experiment 3 is defined to observe the effect of the adaptive change of the workforce. Experiment 3 tests the scalability, robustness, and adaptability of the system. The experiment consists of removing a class of workers performing the NestProcessingTask at a given simulation step and re-introducing it later on. The nest processors are removed at the 20’000th simulation step, which gives the system enough time to start and stabilizes itself. Then, the class of workers is re-introduced at the 40’000th simulation step, and the simulation stops at the 60’000th 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 TETA = <40:50:7>.

**Author’s note:** As GTA’s algorithm might prevent any relevant data from being generated, it is removed from this experiment. In GTA, every robot switch to the task with the highest demand simultaneously (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 robot is removed from the simulation when the 20’000th simulation step comes.

In Figure N, it can be seen that all methods suffer from the change in the workforce at the 20’000th simulation step. RNDTA suffers the least from it as its task allocation system is not based on interaction with the current task demands but rather on randomization. Figure A to-do shows how RNDTA adapts to the situation by distributing the rest of the remaining robot equally over all other tasks. CAITA, DAITA, and PSI’s task completion rates suffer the most from the removal as their curve gets significantly flatter. Although the loss of speed in the completion of the task is reduced, PSI and DAITA show signs of recovery when the group of robots is re-introduced at the 40'000th simulation step, as their curve catches back the lost time. Oppositely, CAITA shows less recovery but still maintains an up-ward task completion trend.

Nevertheless, CAITA and DAITA show clear sign of adaptation in the change of workforce as shown in figure N and M where figure N shows the distribution of labor over the entire period for CAITA and figure M shows the same for DAITA respectively.

Chart, line chart

Description automatically generated

TODO rerun 50:7 exp3☺

Chart, histogram

Description automatically generatedRNDTA

Chart, histogram

Description automatically generatedCAITA

Chart

Description automatically generated

## Discussion and Conclusion

In this paper, a dynamic task allocation algorithm inspired by ants is implemented in a centralized and a distributed way. The centralized version uses an area as an information-sharing center where robots report their current status and are given a new task assignment. Oppositely, the distributed version uses local sharing of knowledge to nearby robots to evaluate the current status of the environment. The algorithm implements a clever system of distribution of labor with a system of energy supplied and energy demand for tasks where as few robots as possible are deployed to meet the task demands.

Moreover, The algorithm assumes a worker can sense its immediate environment through a binary feedback function. This system relates to the Response-Threshold model [link] as the workers respond to stimuli strength. Furthermore, this algorithm is a framework for improvement as it proposes a base of worker characteristic implementation, an open door for non-homogeneous systems and deployments on dynamic environments.

Tests on different levels of communication noise and different communication ranges (3. Experiment on parameters settings TODO) are performed to highlight the performances of AITA in relation to its communication system but also select a bias of noise and communication range that is both challenging and realistic. Then, the algorithm is experimented within a simulation environment in the presence of the communication noise to best re-creates a physical environment. The simulation environment describes a system of three dependent tasks in which tasks’ demands are directly impacted by the initial conditions of the system and the periodical increase of the demand (TODO 3.4 section method or smth). The task allocation method is tested upon five categories that highlight the strength of a swarm robotic system (TODO section): Its robustness, its reliability, its scalability, its versatility, and its adaptability. Moreover, tests over the assumption of the mathematical model’s creators and the energy consumed are conducted.

The Experiments and the performance analysis investigations (section todo) on the relationship of the number of robots, the initial demand of the foraging task, and the periodical increase have shown that the system performances depend on the initial condition as they are closely tight to the number of robots deployed to a task. In order to provide a fair framework of comparisons, initial conditions that best match the task allocation behaviors of the other investigated methods are implemented.

Experiments in section TODO section have proven that, as the distribution of the current state is done dynamically during the run time, the algorithm is adaptive to the change in the workforce (TODO section x.x.y) and the relative demand of the different tasks (TODO section x.x.z). Moreover, the system achieves a noticeably good task completion rate that is comparable to greedy algorithms such as GTA and other insects-inspired algorithms such as PSI, inspired by honeybees, while maintaining a low cost in energy consumption in relation to the covered distance and the number of task switch over a given period (section todo robustness.x). In addition, the experiments conducted on the specific framework created for this paper have proven that AITA can reach a near-optimal task allocation equilibrium given that the number of robots is the same or out-numbers the combined demands of tasks, reducing the squared difference between a task needs and the number of robots assigned to a task to almost null (todo section N).

TODO REFLECT ON THE FIVE CATEGORIES?

TODO reflect. On Zahadat et al from related work?

Although CAITA has shown worse performances than DAITA, both versions have shown strong results in the conducted experiments. However, communication through a single point (information center) is prone to “single-point-of-failure,” as if the central tower falls, the entire system cannot communicate, and the distribution of labor is stuck. Oppositely, the distributed version offers high robustness to communication failures. Suppose one or multiple robots terminate prematurely or fail to communicate their current status. In that case, each robot of the swarm still running will continue to share their current knowledge of the world with others.

This robustness to communication failure is discussed in the next section, where a possible global-world-sharing communication method is elaborated. The effect of single-point-of-failure systems and their relationship to scalability is not discussed. However, XXX’s paper from whom the Chapar towers are inspired offers a brilliant analysis and reflection.

By providing a method of dynamic allocation in relation to the environment, by addressing methodso f bias to counter unintendend task switch, by providing settings on play to address the demand for undefined groups of individual, and by providing a robust communication framework for a centralized and a distributed version of AITA, this paper resolves the challenges introduced in the section 1. Introduction regarding task allocation on dynamic environments.

# Future work

The optimization framework offered by AljenTODO et al. is an overwhelming bed of growth possibilities as mechanisms based on different variables can be implemented to improve the distribution of labor. Much so that if made unnecessary complex, the task allocation problem can become NP-Hard, resulting in an unnecessary complex algorithm. Nonetheless, this section covers relevant future work that can be conducted in order to improve the implementation.

As mentioned in the section TODO, an optimization variable is the location of the different deposit areas within the arena. Indeed, the current implementation makes the robot depose 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 a zone due to their stay-in zone behavior. Mechanisms to improve either the placement of the resources within the area or the detection of resources can thus 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 as for now, this one is not optimal and does not offer the best chance for each robot to escape congestion situation. Pitonakova et al. [link] discuss how congestion affects scalability by limiting communication and information sharing and propose a novel congestion-level-sharing method to solve robot congestion within multi-robot systems.

Mechanisms based on the robot’s location can be envisaged. As the robot moves around the arena, the energy it can supply to a task can be directly correlated with its position in the nest. Robots would have a higher chance of switching to tasks related to their current zone rather than being given one following the task’s stimuli’s strength, noticeably reducing the energy consumed from the system by reducing the covered distance, improving the task completion rate, and reducing the number of task switches per robot.

Another area of improvement is the robot itself, as adding new sensors to the Thymio-II opens new designs in the distribution of labor. As discussed by Alexandre Vanini [link], pheromone imitation methods can stimulate the robot to solve a task, i.e., by increasing its likelihood to switch to a task related to what is sensed by the sensors (that is, the more resources from a specific task are sensed, the most likely the robot is to switch to this task). In other words, there is a play in the hardware to improve the energy a robot can provide to a task given its location and time. In addition to short-range environment sensing, short memory can also 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 high concentration of resources.

The current distributed communication method implemented for AITA, DAITA, has achieved good results in terms of the investigated metrics. However, the system could improve information sharing. Though the system does not contain any single-point-of-failure, losing one robot still means that its information will not further be shared throughout the colony. A global-world-sharing communication system can then be envisaged where sharing the full knowledge rather than its status provides faster global communication and a chance to share failed robot’s information over the colony.

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.

References

<file:///Users/freak/Downloads/Distributed_Task_Allocation_in_Swarms_of_Robots.pdf> -> this one also speaks about big range communication

Cite PSI

TODO si je décris comment les robot detectent et communique peut-être que je peux le faire en ref avec le relarted work

USE **[0] Master’s thesis note. Somewhere as to explain how we think task allocation work**

[1]

-> will have to explain how some model are based on ant characteristic and why I don’t use them

Also cite a bunch of paper and what they did there.

(Aleksandar Jevtić et al., 2012)

<file:///Users/freak/Downloads/Distributed_Task_Allocation_in_Swarms_of_Robots.pdf>

Ducatelle et al.

<https://people.idsia.ch/~frederick/taskallocation.pdf>

Matthiew et al. (Kalra -> ) https://ieeexplore.ieee.org/document/1677943

<https://core.ac.uk/download/pdf/188778566.pdf> -> AUCTION!

<file:///Users/freak/Downloads/paper_preV.pdf>

Payam paper, could be used in the “related work”

Aryo Jamshidpey and Mohsen Afsharchi and in brilliant

<file:///Users/freak/Downloads/Task_Allocation_in_Robotic_Swarms_Explicit_Communi.pdf>

Jérémy Rivière et al.

<file:///Users/freak/Downloads/AnInterruptibleTaskAllocationModel%20(2).pdf>

yongming

<https://ieeexplore.ieee.org/document/4803959>

- Maybe I can slo defend why interaction rate is not optimal because it’s too localisation and time based. ()



- As of now, I have a lot of congestion in my system, maybe talk about how I could prevent it and cite “Task allocation pitonakova”

<https://www.researchgate.net/profile/Lenka-Pitonakova?_esc=publicationCoverPdf&el=1_x_5&enrichId=rgreq-5fec1b5876ed803eb977522816f433ad-XXX&enrichSource=Y292ZXJQYWdlOzMxODM2MDM0NTtBUzo1NzE5MDk4MTU2MzU5NjhAMTUxMzM2NTMzMzMzMA%3D%3D>

[2]

Y. Uny Cao, Alex S. Fukunaga and Andrew B. Kahng, “Cooperative Mobile Robotics: Antecedents and Directions”, Autonomous Robots