An Ant-Inspired Task Allocation Simulation Implementation for Swarms of Robots

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

Payam Zahadat (90%)

Kasper Støy (10%)

# Abstract

Intro on swarm and task allocation

This project presents the a distributed method for task allocation inspired by ants and their social behaviours.

Say also that is intends to explore task allocation in swarm 3 independent task blahblah

This paper proposes a software implementation of an ant-inspired mathematical task allocation model created by Alejandro Cornejo et Al. [link]. Moreover, it declines the implementation in two architecture of the swarm robotic paradigm; A centralized and a distributed version. (say that the decentralized one does not require global knowledge ) Through a set of experiments and thorough comparisons against strong task allocation methods, is has been proven that .. Also speak about PSI RND and GTA More……

# Introduction

Nice for intro : <file:///Users/freak/Downloads/An_Extensive_Review_of_Research_in_Swarm_Robotics.pdf>

Talk about interdependent tsk as done in brutshy

Swarm robotic can be defined as « a new approach to the coordination of multi-robot systems which consist of large numbers of relatively simple robots which takes its inspiration from social insects. The most remarkable characteristic of swarm robots are the ability to work cooperatively to achieve a common goal. ». [1]. The studies of task allocation systems related to swarms of robots is mainly used to understand how a complex system of task can be solved using multiple agents [link 2-4 symmetry]. The tasks can either be achieved by an individual, alone, or by a group or robots, cooperatively [link?]. Existing practical applications of swarm robotic includes … LOOK FOR EXAMPLES.

Distribution of labor in groups of individuals can be considered done in two ways: Firstly, given that the environment is limited in its growth, one can implement pre-determined task allocation strategy, such as robot-planning [LINK], where the individuals will be able to perform a set of given task with time and precision. The second is task allocation is dynamic environments where robots are given a set of behavior-based mechanisms and the task allocation strategy has to be adaptive. The main challenges faced by swarm robotic when dealing with dynamic environment are the following:

* Coordination of a large groups of individuals is exponentially harder 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
* Inaccuracy and inconsistency in dynamic system can exponentially grow if non-resolved, resulting in a high energy consumption.
* Scalability and task allocation dynamicity is hard to achieve through limited strategies

The proposed ant-inspired task allocation implementation hops to resolve the beforementioned challenges by:

* Dynamically allocating individual considering the current environmental demand.
* Reducing inconsistency in task switch, thereby reducing energy consumption.
* Being adaptive to a large and un-defined number of task for an as large of a group of individual as possible while keeping accurate distribution of labour.
* Creating a solid communication framework robust to communication failure and light

**Maybe move as I first speak about comm an information center in the related work section? -> could turn this paragraph into inspiration from related work**

Moreover, this project intends to provide a framework of precisely defined environment and experiments description for everyone to reproduce the work and improve the implementation or compare it. To address the mentioned improvement, this paper explore the implementation of an ant-inspired algorithm via two architectures of the swarm robotic paradigm through the implementation of …. Firstly, the centralized one (see figure n), where the information about the environment is not shared among all the individuals but is rather 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] but has obvious downsides when the group of robots become larger as the communication failure (information loss) and overhead quickly creates a disturbance in the system [link]. Moreover, this system has what is commonly referred to as a single point of failure, where if the information center breaks or stops functioning, the entire swarm is impacted and cannot perform further action [link]. The second architecture is the distributed architecture. This time, the information is shared among all individuals through local communication where 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 of 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 is only locally to one robot, which does not impact the rest of the swarm.

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

As said in the paper, they provide a framework that let us choose ant feature and more

**Maybe that a bit bold of an assumption to say that I want to explore that, then I need to discuss it big time in the discussion, also because I kind of let AITA apart**

TALK ABOUT THE RESPONSE TRESHOLD?

- Also, my system AITA was not implemented ANYWHERE (on the web that is, no paper has done it) so it’s nice that I provide a proof

My thesis is also an experiment proposal so that hopefully people can use it afterward with their own system. Also, it is meant to be use for overly simplistic robot as ants.

Explain what I will do and from who I will “copy”

**Say something about task allocation .. swarm .. the future .. refer to other paper, they might’ve the answer**

Talk about response treshold

**I think mine is treshhold as well.**

**[1]**

**“un mot qui veut dire avertissement sur »**

*#! I know I want to use robot simulated because I want to assess the efficenicy of the allocation system for robots. Doing it with few robot wouldn’t prove so much.*

<https://www.frontiersin.org/articles/10.3389/frobt.2020.00036/full> -> this website explain why is swarm robotic. Maybe I can have a reflexion part on the thesis, and transpose it to the ants. Like it says

“ **Group size regulation** allows the robots in the swarm to form groups of desired size. If the size of the swarm exceeds the desired group size, it splits into multiple groups.” But ants don’t have such complex system.

**[0]**

## Related work

Models that tries to solve dynamic task allocation have already been experimented, 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 tasks demands play a role in the task allocation distribution. The model have shown weaknesses in the distributions 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 great scalability in terms of the number of robots and number of tasks. **Ducatelle et al. [?]** propose two methods for decentralized concurrent task allocation of groupd of flying robots in a confined area where novel communication methods are used for globally share tasks demands. The first method is a interaction through light signals based communication, where the robot react to different light signals and colours dependent of the current task distribution. The second is a gossip-based method where the infrared sensors present on each robots allow primitive communication of task status and robot status. The results concluded that a gossip-based communication using infra-red sensors was more efficient than using other kinds of sensors to detect emissions of light by other individuals in limited environment. However, they also concluded that the gossip-based communication was subject to packet loss and therefor might be less scalable. **Matthiew et al. [?]** propose a distributed algorithm for task allocation based on the work of Kalra et al. and their auction-based method. **A. Jamshidpey and M. Afsharchi** propose a study of multi-robots 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 remain unchanged until it runs out of *search energy*. The second one is a improved version of the first method called the *dynamic communication-based approach* where when the a robot runs out of energy, it can chooses its next time following a probabilistic model. The remaining methods (a decentralized and a centralized) uses Chapar towers (radio communication stations) placed in the environment and that act as information center. In the Decentralized version, groups of robots called the “Chapars” will transmit information from the Chapar towers to the workers and so forth. In the centralized version, one Chapar station covers the area and robot report to it. The conclusion drawn is that there is not a significantly higher efficiency when comparing the centralized or decentralized version, but point out the single point of failure and robustness issues in communication that may arose. **Jérémy Rivière et al**. [?] propose methods of self-awareness in robot task selection based on the *Response Threshold Model* (LINK TODO) where individuals are able to decide whether to pursue a task or to stop it independently of the task stimuli. The results have shown that the swarm is able to each 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 environment 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 Treshold Model* that does not utilize any communication medium. This research have shown that such algorithm can successfully adapt to the demand and task completion growth via a solid labour distribution regardless of explicit communication within the individuals of the swarm. **Finally, Arne Brutshy et al** [?] propose another communication-less task allocation methods based on interaction rate within individuals using a set of interdependent task, and prove that the method can reach near-optimal task allocation.

# Ant-Inspired Task Allocation Algorithm

This section describes the main algorithm used by the swarm of robots to achieve task allocation, and how it is integrated to the simulation environment. Throughout the section, the terms AITA, CAITA, DAITA will be used. AITA stands for **A**nt-**I**nspired **T**ask **A**llocation and is the implemented algorithm, CAITA stand for **C**entralized **A**nt-**I**nspired **T**ask **A**llocation (a centralized version of AITA), and DAITA stands for **D**istributed **A**nt-**I**nspired **T**ask **A**llocation (a distributed version of AITA).

## Introduction

Ants are great social insects in many ways. From their adaptability to a great range of environments, to the extremely high level of self-organization in colonies reaching to sometimes up to 10 millions of individuals, they are a great source of inspiration for robotic and swarm intelligence. Their most interesting feature is their maximization of the distribution of labour 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 behaviours, which at the time this paper is written (Q2- 2021) has not been implemented nor tested in a simulation environment. This project is an attempt at giving this algorithm a fair implementation and experimental comparisons against other task allocation methods. The algorithm is inspired from the response threshold strategy where the robots react to a 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 rather can tell whether or not a task needs more attention.

## 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 being the **demand** for a task t at a given time T. The **demand** can depend on many aspect of the current environment, such as weather conditions, the location of the colony, the current number of ants in the colony, etc. 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, but also on 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 being 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 label of much is 1 energy.*

## Assumptions

Alejendro et al. (nametodo) define a satisfying task assignment as being 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 make the assumption that if one task can be set to an equilibrium, then all other task will be served, because when the equilibrium is reached, the surplus of robot assigned to the task can be re-assigned to potential task in energy deficit.

## Model restrictions -> does this section really 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. It is because the environmental variables, ant’s characteristics, and experience settings are so phenomenally broad that it is impossible to include every one 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 which all share the same skills and characteristics to solve a given task. Sharing the same abilities to solve a task is important as it 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 the others. Moreover, since the simulated environment is controlled and not meant to reflect hard environment such as a jungle or a desert, but rather 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 memory of past experience, which means 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 the system where the energy an individual can supply to a task is dependent on its characteristics is furthermore discussed in the section “future work”.

## Task allocation algorithm

This section is an overview of the AITA algorithm. For a more in-depth presentation of the algorithm, please refer to the original Alejandro Cornejo et al’s research paper.

### 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 in energy surplus, -1 otherwise (not that the binary feedback function does not provide enough information for the workers to tell whether a task has reach 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 is unable to quantify by how much a task is in energy deficit or surplus. This is very important as it fundamentally shapes the way the task allocation system works. Being able to sense only little and local information is also biologically accurate as ants don’t have the exact knowledge of the energy demand of a task [link] in the sometimes up to 1’000’000 million individual nests [link].

### Algorithm

The algorithm (which considers the size of the colony |A| as being fixed) works as follows: Firstly, all worker executing the algorithm maintain 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 which is used to determine which task a worker will be executing 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. Worker first start idle and in the Resting state, and as the environment evolves, they fill up a candidate list which contains task with 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 Resting state (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 Resting state 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 Resting state 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 N depicts a correction made to the algorithm, explained in the section “correction on the original algorithm TODO”.

## 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 for the following reason: When the value for a task in table Q is 3, it is added to the candidate list. However, it is not given that this task is selected as the worker selects a new task with probability ½. This means 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 and the robot is stuck without any assignment until eventually a task is in energy surplus and the worker assigns its value to 0 in 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 important 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 algorithm behind the communications of the centralized version and the distributed version of AITA.

### Feedback function

The feedback function is at the core of AITA as it allows a worker to access knowledge about its current local situation. The work explored in the related section is a great source of inspiration to design how the information is accessed by the individuals. One can use Interaction Rate Methods where the task allocation algorithm is based on the response threshold associated to a task and sensing of the local environment[yongming]. However, this kinds of methods relies on the population size of the swarm as the lower the population is, the lower and less explicit the stimuli associated to a task can be, which is problematic in an algorithm that shall not depend on the colony size to be performant (reflect on that somewhere). Oppositely, one could also assume light sensors are placed on top of each agent which allows light signal-based communication (todo link). However, methods relying on directional sensors and explicit signal discovery are less accurate and are more prone to noise and faulty behaviour (todo can I find a source for that?)

Instead, communication via transmission tower is chosen such as in the work of … where Chapar towers are placed on top of each agents and can be used to share and receive information.

* -> **Aleksandar Jevtić et al. Also zigbee communication line and reflect upon chapar towers**

**Centralized and Distributed**

The beforementioned algorithm has been implemented in both a centralized fashion - where a communication area (the nest) severs 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**

**TODO: make sure I say CAITA robots does not know about the current demand, they just share how much they have done since the last report**

The centralized ant-inspired task allocation implementation or CAITA works as follows (as shown in Figure N): At their initialization, each robot is given the initial demand for each task which is defined by an outside person. The robots then have to perform a set of 3 dependent tasks (more on this in section TODO) while periodically reporting their status to the information center (Note that: a robot is not being assigned a new task if it carries a payload). Each time a robot reports its status, it shares its identification number, its current task, its current state, and what it has been working on since the last report. Once the information center receives these pieces of information, it updates a table of global knowledge, decides on a new task assignment for the robot, and send it back to it.

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 and 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.

Talk here about shift error knowledge if not introduced further down?

TALK HERE about why its communication a not something else? Relate to the one who does not use communication and to the one who does

**Distributed AITA**

Significant changes are made in the distributed version of AITA. This time, the memory is not contained in a single place but rather shared throughout the entire swarm. To do so, each worker holds an array of demands initialized at the start of the simulation. Along with this array, each robot has a memory of every other robot’s state. At each simulation step, every robot tries to broadcast its current status – I.e. its identification number, current task, state, and its current progress on each task. As another robot receives this information, it updates its memory on the robot and runs the AITA algorithm to be attributed a task.

Robustness mechanisms similar to the one implemented for CAITA are also present in DAITA. This time, each robot keeps track of the last time it has been contacted by a robot. If this elapsed time goes over a specific pre-defined time, the robot is considered gone from the other robot’s system.

# RND, GTA and PSI

**other chapter)**

## RND, GTA and PSI

To give a more in-depth analysis of CAITA and DAITA, it has been chosen to implement three other algorithms that would evolve and be experimented on in the same environment with the same set of tasks to perform. This section provides an overview of each of them, highlighting why they have been chosen over others and what can be expected from every one of them.

### Random Task Allocation Algorithm

The random task allocation algorithm, or RND, is a method where each individual, given that it is not currently performing a task (currently performing a task means currently carrying a payload), is attributed a new task every 600 simulation step (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 an other). Since the robots do not require sharing any kind of information and don't need to be aware of the current world state, this task allocation system does not suffer any kind of communication failure or overhead, which makes it highly scalable and robust.

This algorithm has been chosen because it is the simplest algorithm one can design, and thus 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 worry about the current world's state and is very inconsistent (for instance, even though the probability is extremely low, one could end up in a system where only the foraging task is performed throughout the entire lifetime of a simulation).

### 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. As for the CAITA, DAITA, and RND, a robot cannot be attributed to a new task if it is currently performing one. Because the information is distributed among the entire system (the memory and communication systems are the ones used by DAITA), this algorithm suffers the same challenges as DAITA, that is, communication failure and system disturbance.

A greedy algorithm has been selected because it perform well in a wide range of situations, but consume a lot of computational power. Thus, this algorithm is expected to perform well but at higher computational costs compared to other systems.

### Partitioning social inhibition Task Allocation Algorithm

The Partitioning Social Inhibition task allocation algorithm, or PSI, is a system issued from a 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." Being adaptive to changes in the swarm size and demand for the tasks is important because it is part of what the other TAs are being experimented on. PSI is also interesting because it does not fall under the category of algorithms that are easily applicable to any kind of situation, rather they are robotic related or not (such as GTA and RND) since it is also an attempt at solving the task allocation problem. This sub-section is an overview of the algorithm and the way it has been implemented within the system developed for this project.

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 is executing 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, but the system is also expected to suffer from the consequences of applying the algorithm to the author’s environment (which is elaborated on under the discussion section TODO).

Each robot of the swarm holds an x value that represents their physiological age – A value that represent the biological real state of a living things (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). PSI aims to distribute each individual's x value relative to the current demands for the tasks to achieve equilibrium (recall that the equilibrium is when the number of robots assigned for a task matches or covers the current demand of a 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 member of the swarm, but this paper does not intend to cover that. For further information on how the local interaction within individual changes the x values, please refer to the original paper.

As mentioned above, PSI has not been written to run under the same kind of environment as for the one DAITA and the other algorithms have been designed for, meaning that PSI has been adapted to fit this project's simulation and its constraints. 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 a demand for a task can variate from -inf to inf, whereas in Payam et al.'s system the tasks have a value that represents a fraction of the current demand and usually variates between 1 and some positive number. This means that in the PSI's original system, no task can have a negative demand or a demand of 0. To counter that, it has been chosen to map the actual demand of the environment to a 1 - 20 scale (only for PSI) as follows:

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

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

As for all the other algorithms, PSI cannot attribute a task if the robot is currently performing one. What it means for PSI is that the value x of each individual is delayed as long as the worker is currently carrying a payload. As tests have shownt, it does not impact the efficiency of PSI.

A wide range of tests and experiments 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.

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

In order for anyone to reproduce this paper’s work, this section quickly defines PSI task allocation system includes some variables that one can act on, table N defines what values have been chosen for each variable. More information about what each variable mean and the what influence it has on the PSI system can be found in the original paper [link].

TODO add table.

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

# Experiments

This section precisely describes the set of variables and assumptions used for the system and the experiments. Moreover, it shows and discuss the results of the experiments conducted over the different parameters of the simulation and the different methods of task allocation.

TODO: say somewhere that each resource can be consider as a sub-task?

## Environment framework

The different task allocation methods, CAITA, DAITA, PSI, RND, 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 4 types of agents (idle, foragers, nest processors, and cleaners), a nest including 3 main areas (or chambers), the dump area (in blue, 1.4 meters x 1.4 meters), where resources collected from the outside world are stored and where robots that use the CAITA algorithm and the centralized communication report their status. The transit area (in pink, 1.4 meters x 1.4 meters), where resources processed from the dump area are stored. Finally, the waste area (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 3 areas is considered a foraging area where 2000 food 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 effective performances of the different TAs). The topology of the world is a rectangle box bounded in all its directions that 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. to 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 any 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 its colour 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 3 dependent tasks which means that the demand in the foraging task always influences the future need in the 2 others. In other words, for the demand to raise 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 raise 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 like tasks such as collecting warehouse supplies and carrying them out to other parts of a hangar in a transit area to be processed. Note that: this set of task is not representative or limiting the application of the AITA algorithm, but is only highlighting the performances of this algorithm compared to other methods in a specific environment. The algorithm is expected to be as performant with a set of independent task.

Scatter chart

Description automatically generated

* Maybe have a better one that shows also every kind of resources and robots and less points (like 500) (colors).

## Environment setting

In order for the experiments to be as fair as possible, and to make sure each task allocation method is given the same chance to succeed, the environment is framed with a set of assumption an variables. Firstly, the environment 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 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 is able to somewhat precisely move to a given coordinate in the plan during 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 at the same time. As for CAITA, The DAITA system assumes an instant communication transmission time when sharing its information to someone. Moreover, each robot is only able to receive one transmission at a time (that is, one each simulation step), dumping any other incoming message meanwhile. In order to simulate this behaviour, a random transmissions receiver mechanisms 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 both use the DAITA communication system suffers the same communication restrictions and challenges.

### Mutable variables

Along with the set of assumption, a set of specific variable can be changed prior to run a simulation, and it is expected that each of these variables have a high impact on the system’s performances depending on their 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 foraging task is a fix value set ahead of all experiments. This parameters 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 experiment as fair as possible for the following reason: The AITA algorithm makes sure that the demand for a task always meets equilibrium (given that enough workforce is available). This means that the system always try to deploy as few robots as possible to achieve a task, whereas system such as RND, GTA and PSI always distribute all the workers over the set of task. This creates an disbalance in the comparisons. By periodically increasing, 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 on dynamic environments as the demand in the different tasks is constantly changing.

As most of the experiments are run with 40 robots, tests have proven that increasing the demand by 7 resources at every 500 simulation step is a good number 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 of the increase), over the whole period.

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. And finally, the reliability of the system stating that the robot should be consistent in its probability of solving a given task. Along with this 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, a in-depth performances analysis of their energy consumption is conducted.

In order to evaluate the performances of the proposed methods upon the defined categories, different metrics are taken into consideration; The task completion rate, the total covered distance for all robots, the distribution of robots in the different classes of workers, the number of task switch per robot and the demand and the energy supplied for each task. For systems that have a distributed communication (DAITA, GTA, and PSI), the average sensed demand by each worker is also compared to what the real demand for each task is. Each experiments is run 5 times and averaged to give the more fair outcome compared to a single run.

## Experiments on parameters settings

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

### Noise (here nice past tense because it’s prior to the exps and results in time)

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. Both CAITA and DAITA use the following system for this experiment: 40 robots are used, the task is to collect and fully process 150 resources 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 finally reach Pnoise = 0.99.

**Noise DAITA**

As can be seen in figure N (todo), the different levels of tested noise don’t show any kind of significant variation in the task completion rate (the rate at which the given task is completed). Nonetheless, a small 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 where it can be seen that both Pnoise = 0, Pnoise = 0.3, and Pnoise = 0.7 have around the same error, reaching low fluctuation with an averaged error of TODO N, X, A for Pnoise = 0, Pnoise = 0.3, and Pnoise = 0.7 respectively. These slight variations even at high noise level such as Pnoise = 0.7 are plausible and are the result of the communication system implemented in DAITA. In the DAITA communication system, with each simulation step, every robot broadcasts its current knowledge of the world to all of the others individuals. 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 packet each simulation step, or less than 1 each round which means that successful communication can take up to on average 2.56 rounds to happen. This can be seen in the swarm’s perception error e of the actual environment shown in figure N were when the noise reaches a high level, the error skyrockets. Nevertheless, given enough time the error e plummets and the system stabilizes to reach just above the same error e as the other, or XX todo 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 in communication failure reaches a high level, the success rate can become so low that the swarm struggles to agree on 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 – Which means high oscillations in the task allocation resulting in a high number of task switch per robot. Inversely, for a high probability of communication failure, the swarm struggles to adapt quickly 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 high probability in communication failure, also called *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) as noise slows down the rate of information sharing, leading the swarm to over-collect resources. This also explain why for Pnoise = 0.99 the curve slightly variates in Figure N and the system finishes before the others. As the *shift* is big, the system struggles to adapt and fails at optimal labor distribution and over-collect resources, leading the swarm to take a time advantage over the other noise system.

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**

In the case of CAITA and unlike DAITA, the task completion rate graph in Figure N clearly shows signs of weakness when the level of noise Pnoise reaches 0.99. However, the other level of noise seems to achieve the same performances. Recall that the communication system of CAITA is fundamentally different that the one implemented on DAITA. In CAITA, each worker have to report to a central which then distributes tasks. This means that there’s twice as much more chance that a communication between a transmitter and a receiver fails as the robot first have to communicate, and then the nest send out the new task assignment.

Chart, line chart

Description automatically generated

In conclusion, both DAITA and CAITA have shown high level of robustness to communication failures. Furthermore, even though the higher the probability of communication of failure is the more “messy” the knowledge error is, both systems seem to be able to compensate over time this problem and are expected to keep a healthy system for long periods of test. However, if the system is highly dynamic (lots of changes) it is expected that systems with high noise is be performing worse than system with low noise as system with high noise cannot adapt fast enough to the situation, leading into the knowledge error shift to be incontrollable, however, not covered in this paper (as shown in figure N, TODO where the system increases its foraging demand of a high random number from 5 to 15 every 500 hundred simulation step).

**Chosen Pnoise parameter**

For the rest of this document, each experiment assumes a noise level Pnoise of 0.3. This noise level still offers consistent results and a fast adaptation to the environment and remains a high number of communication failures, which should be a proof of the robustness of AITA.

### Communication range

The influence in the change of communication range has also been tested before the experiments to observe and highlight the effects of it on the system. The test 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). Moreover, the system assumes that there is no noise in the communication. The test consisted of using DAITA (workers in CAITA can share information as soon as they are on the communication area) with 40 robots and make it collect 150 resources as fast as possible WHAT IS THE DEMAND. The range is first of 10 centimetres and increases to reach 13 meters, which covers the entire arena. The result are shown in Figure N.

Chart, line chart

Description automatically generated

Among all the experimental settings tested, the task completion rate of range 0.1 is significantly higher than the others, reaching 1 20’000 simulation step later. Ranges 5, 7 and 13 meters offer the most consistent result as their task completion rates are similar looking, whereas ranges 0.5 and 1 meter show slight variations and are the fastest at reaching 1. As for the section NOISETODO, this alone is not enough to demonstrate the consequences of short communication ranges over different task allocation methods. This is why the swarm’s perception error e of the actual environment has been recorded and is shown in Figure N.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

As can be seen in figure N, the ranges of 0.1 and 0.5 meters offer the worst speed in environment adaptation resulting in a high swarm’s perception error whereas ranges over 1 meter (included) offer a low level of error. This can furthermore be seen in the number of task switch per robot shown in Figure N. Systems that have slow adaptability to the environment have fewer task switch as the need for tasks struggles to be shared globally, whereas robots with long-range communication access the information faster resulting in a higher number of task switch per robot as the workers can quickly adapt to the dynamic environment.

The effect of a short-range communication distance can be correlated to the *knowledge error shift* from the section NOISETODO as the shorter the range is, the slower the robots are notified about changes in the environment. This effect explains why short-range communication systems demonstrate a faster task completion rate than long-range communication systems in Figure N, as

As mentioned in the “NOISETODO” section, a direct consequence of a high *knowledge error shift* is the over-collection of resources. This effect on different ranges of communication can be seen in figure N or figure A (todo FF0) for range 0.1 and .5 meters respectively, where the demand sensed by the individuals is once again shifted from the actual demand of the environment. Over short periods, over-collecting can be harmless as the distribution of labor is good enough for the swarm to process the collected resources. However, long-period simulations, over-collecting becomes harmful as the swarm struggles to quickly adapt to the environment and get overwhelmed by the amount of resources it has to process. As the range of communication gets longer, the problem fades and almost disappear resulting in no over-collection of resources.

A solid feature of the environment that is directly related to a healthy system is the periodically visible 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 of the swarm adapting to the demand. The first two graphs are the labor distribution of the range 0.1 meters and 0.5 meters respectively where it can be seen that as there’s no spike, there’s no sign of the swarm adapting to the periodical increase of demand in the foraging task, which direct consequence is a sign of a high *knowledge error shift*. Inversely, the two other graphs show the distribution of labor for range 7 meters and the entire arena where clear signs of swarm adaptation are seen as periodical peaks are present throughout the entire period.

|  |  |  |  |
| --- | --- | --- | --- |
| 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**

From now on, each new experiment assumes that the range of communication for DAITA, GTA, and PSI covers the entire arena. Indeed, as ranges below 5 meters suffer inconsistency and a high *knowledge error shift* they are disregarded. As for ranges of 5 and 7 meters, they do both offer consistency and fast environment adaptability, which underline that using a range that covers the entire arena will have the same effect but will offer the maximum consistency and accuracy in experiment results.

## 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 |
|  | 7 | *ForagingTask* demand increase |
|  | 500 simulation step | Period at which increase *ForagingTask* demand |
|  | See experimental settings | ForagingTask initial demand |
|  | 0 | *NestProcessingTask* initial demand |
|  | 0 | *CleaningTask initial demand* |
|  | See experimental settings | Number of robot |
|  | See experimental settings | Number of resources to collect to complete task |

*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 individual can complete a given task. Moreover, task completion rate can be used as a optimization goal as the sooner the swarm complete the task, the better the system is. In order to experiment the task completion rate, experiment 1 is defined. Experiment 1 consists of collecting, processing, and cleaning 150 resources as fast as possible [NDPDB], PARAMETERTODO is set to 50.

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

The first set of tests performed on the CAITA and DAITA systems with the number of robots 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. Furthermore, the trend shows that in a given environment, it is not given that the task completion rate gets higher as the number of robots grows. Indeed, both CAITA and DAITA demonstrate struggle when the number of robots goes above 50. In the former, using 70 robots or 100 robots is barely as good as using 50 robots as they achieve the same performances. 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 than 40 robots and 50 robots. A Part of this drop in performances as the number of robots increases is due to 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 they are blocked by others. Furthermore, congestion implies that optimizations of the task completion rate by improvement of the positioning of the different areas and the implemented de-congestion algorithm are possible, however not covered in the experiments, but rather reflected upon in the discussion part. 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 many robots as needed on a task to reach equilibrium (when the energy supplied to a task is the same as the demand of the task). Seeking equilibrium means that in a world where there exist 70 or 100 robots but the demand is only of 50 resources the algorithm will only deploy 50 robots to cover it. This behavior can be thought of as being “underperforming”, but it just is 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 furthermore elaborated in the next chapter.

?.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***

TODO maybe don’t call it a ratio

The relationship between the number of robots and the initial demand of the *ForagingTask* can be written as follows: N*robots*:ID*foragingTask or TETA*. This relationship is tightly bounded to the fixed periodical increase of the *ForagingTask* demand discussed in the section (TODO NAME OF SECTION). Indeed, if no improvement in the task completion rate can be seen with 70 robots or 100 robots, it is because a lot of the workers are idle as the workload of the environment is not high enough (see Figure N). Inversely, Figure A shows a system where TETA is good enough so that the entire system remains busy at all time (TODO). The definition of the relation can then be rewritten as follows: <Number of Robot:Initial Demand ForagingTask: Periodical Increase ForaginTask Demand> where *Periodical Increase ForaginTask Demand* is a fixed value.As one increases the number of robots, it also has to increase the initial demand of the *ForagingTask* so that the system keeps a specific business so that it can increase its task completion rate. To furthermore observe the effect of the relationship settings, the series of Figure N to A shows the results of the parameters settings for <100:25-50-75-100-150-200:7> using DAITA (GOALPARAMETERTODO is set to 150).

Chart, line chart

Description automatically generated

As can be observed, the better the parameter settings become, the higher the task completion rate is. This concludes a small experiments on task demand parameter settings where from now on, 40:50:7 is the chosen parameter as it keeps the system busy for the entire period giving it a chance to show its best performances against the other task allocation methods.

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

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

The same experiment is conducted on RND, GTA and PSI with 40 robots and compared to the two previous systems. Among all the experimental methods, Figure N shows that with final of 10024 simulation step, GTA achieves the best task completion rate than any other system finishing a short while before DAITA (10262 simulation step) and PSI (10389 simulation step). DAITA is as performant as PSI whereas CAITA (10935 simulation step) is slightly off compared to two distributed systems. RND (12536 simulation step) shows clear sign of under-performance.

Chart, line chart

Description automatically generated

## AITA: Robustness and Energy consumption

Experiment 2 is defined as a robustness experiment designed to show the performances of the different task allocation methods over a period of 30’000 simulation steps. This section tests the assumption of ALEJENDROTODO that AITA is made to reduce the squared difference between the need for a task and the number of robot assigned to it (all tasks combined). Moreover, it looks over two optimalization goals to reduce energy consumption; the total covered distance of all robots, and the distribution of the number of switch per robots.

**Robustness of the different task allocation method**

Figure N shows the squared error for the different task allocation methods. It can be observed that methods which consider the demands in their task allocation algorithm do highly better than the one who does not, such as RND. RND reaches an average of 41484 squared error and the upward trend shows that this number is not falling in the near future. This is due to RND disregarding the current need of the environment and rather distributing equally its workforce over the three tasks, leading it to over-collect resources (behavior discussed in section TODO, see graph TODO), and thus leading to an increasing error as the demand for each task is not met.

Chart

Description automatically generated

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

PSI, with a averaged squared error of 99, CAITA, with a averaged squared error of 96, and DAITA, with an averaged squared error of 44 show a similar trend with a first pick as the overflow of resources overwhelmes the system. Once this flooding of resources is handled, the system tend toward an almost null squared error, leaving only the peaks of periodical increase in the *ForagingTask* as a variation in the curve. GTA shows the same downward trend as DAITA, CAITA, and PSI, however, its averaged squared error over the period is of 186 as this algorithm underlines high inconsistency in task allocation, meaning that overtime it will perform worse than DAITA, CAITA and PSI. Moreover, it is not diminishing at a fast enough pace to be performances compared to the other TAs over longer period of time.

As CAITA and DAITA reach a low squared error over time, it proves that ALENJENDRO TODO’s assumption is correct. Given enough time and a not to challenging environment, AITA can reach task equilibrium.

**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 travelled 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 the 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 respectively and in increasing order of covered distance. In comparison, RND 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. However, this is only efficient for DAITA has its communication system is fast enough for the swarm to adapt to the change and deploy as few workers as needed. CAITA having covered a significantly higher distance than DAITA is a clear sign of a weakness in the information-sharing center and swarm adaption to dynamic environments.

**Optimization result on task switching**

The number of switch per robot is also an optimization goal that each TA should minimize as the lower the number of switch is, the less disturbance there is on the system. Figure N shows the number of task switch per robot for each system. It can be seen that PSI has the highest number of task switch per robot compared to the other task allocation methods, with a total of 1774 task switch in average [NDBDP]. The other methods do not show extensive number of task switch over the experiment, however, CAITA is the methods that performs the best with an average of 10 task switch, followed by RND, GTA, and DAITA with an average of 29, 44 and 56 task switch per robot respectively.

[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 workforce

The adaptive change in workforce is meant to observe the speed at which and how well a group of individual can adapt to a sudden change in the number of workers performing a task. The adaptive change in 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). To observe the effect in the adaptive change of workforce, experiment number 3 is defined; Experiment number 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 timestep, and re-introduce it later on. The nest processors are removed at the 20’000 simulation step, which gives the system enough time to start and stabilized itself. Then, the class of workers is re-introduced at the 40’000 simulation step and the simulation stops at the 60’000 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.

As the goal of this experiment is to observe how robots adapt to a change of workforce, GTA is removed from this experiment. Indeed, in GTA all robot switch to the task with the highest demand, it means that usually all robot are in one task (it may happened that, due to communication delay, a robot changes later than its neighbors). Moreover, there is a likelihood of 2/3 that no robot is performing the *NestProcessingTask*, meaning that no robot would effectively be removed, preventing any relevant data to be generated.

The comparisons of the adaptive change in workforce on different task allocation system is run with 40 robots and shown in Figure N. Figure N

shows that the general trend shows that each TA suffers from the removal of individual as 20’000 simulation step and 40’000 simulation step clearly show signs of change for each curve. RND suffers the least from a change of workforce, highly explainable as its task allocation system is not based on interaction with the current environment situation but rather on individuals will to select a random task. Figure A todo shows how RND adapts to the situation, just by distributing the rest of the remaining robot equally over all other tasks. DAITA and PSI do not highly suffer from the lost in their workforce, and as can be seen they are able to heal from it when robots are re-introduce, as their task completion rate goes back to normal. CAITA however, seems to suffer more from the loss of individual as even though it was as performant as DAITA and PSI, its task completion rate drops and never catches up with the two other. 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 generatedRND

Chart, histogram

Description automatically generatedCAITA

Chart

Description automatically generated

## Results

* Talk about the ratio
* . Jérémy Rivière et al je peux m’en inspirer de leur conclusion pour parler ue c’est independant des condition intiale ou blahblah

(TODO maybe move it in the discussion area) SQERROR This shows that both algorithms are reaching equilibrium in their task sooner or later, minimizing their squared error

* **Talk about task allocation, demand and how energy supply could’ve and could impact the result**
* 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”
* I imagine it is going to be interesting to assess how many robot it needs for a set of task to be at an equilibrium -> I cam reflect on that
* Say that psi was not designed for such task, which will explain a lot of thinngngngnggs

As mentioned in the section TODO, an optimization variable is the location of the different deposit area within the arena. Indeed, the current implementation makes the robot deposed their payload in the center of each zone which drastically increase the time-to-find a resources as robot will usually crawl at the edges of a zone due to their stay in zone behaviour. Mechanisms to improve either the placement of the resources within the area of the resources detections can thus be implemented to improve it. Moreover, congestion also discussed in the same section can be something one can improve. The implemented robot avoidance behaviour is not optimal and thus does not offer the best chance for each robot to escape a congestion situation.

Mention Aryo Jamshidpey and Mohsen Afsharchi and their single point of failure section (or create mine)

*#! ob: when more demand than robot, no oscilliation*

*#! ob: when too much osc the robot struggles to complete a task because it is always pulled somewhere else.*

## Discussions / Conclusion -> sort of

TODO: Make sure that there is a clear winner for each, reflect on it in disuccsion

Looking back at what was expected from the AITA algorithm, few conclusion can be drawn,

* Say that in system where there’s no increase of the demand AITA will be extr efficient by allocating only the good amount of workers.

Here faire un retro de toutes les exps et dire genre (ouais bah si GTA était bon la en revanche il est nul la .. »

**Discuss possible variable and how they have impacted the task allocation model**

1. the paper proposes initial condition (such as no mouvement in task needs for a define amount of time) -> maybe I could propose stress test to relate to real life condition

-maybe say here that the robustness has been proven by the noise experiment?

For the conclusion

Maybe the greedy woul’dve perform worse with more task? Take the experiment from which I stole all the graphs idea because they were comparing GTA over 100 task and with other TAs and it was the worse

* GTA first -> It can be that the system in not challenging enough­
  + Also its trend is more or less to have these weird flat line each time
* Say somewhere that RND and GTA and PSI overperform as they don’t have mechanisms that prevent them to no allocate robot if task is good

# Future work

# The paper says it could also be impacted by previous experience .. maybe the robot can have a short memory

# That would say "ho .. I was close to food 10 timestep ago.. it is likely that I still have food nearby"

# The robot cannot sense their long-range environment, but maybe, for task such as food, we could sense the short # environment and say "if I sense food then the energy I can provide is higher"

# ? maybe .. if the robots know about any last foraging point .. then maybe the energy it can supply is greater?

#! or .. if you already are on the area for the task .. maybe increase?

In realf life it would be different, maybe reflect here with some paper from Gordon

**Possible improvements to the math model**

* **Share your entire memory to a robot instead of just your understanding, making it even more robutst.**

Given the exact implementation of the math model, the experiments have reach a certain result, but it could be changed so that ..

* Try to use different feedback method function based on different communication medium

1. Every n step, re assign every robot with the current world state -> my take is that the distribution is going to be better
2. maybe the robots could "see" or "reassess" the needs when entering an area or something .. idk (even though it says that they would not try to assess how much energy can an ant give to a task as it makes the problem NP-complete)
3. maybe the gordon idea with the map could be tested as improvement (even though it says that they would not try to assess how much energy can an ant give to a task as it makes the problem NP-complete)
4. sometimes some ants are not even allocated .. maybe if all task have enough ants some idle should be attributed a random task=

Also the real-life implementation

Read payam’s email, I think she mention somewhere something that could potentially be future work

Conclusion

[small re-introduction?]

In this paper, a dynamic task allocation algorithm inspired from ants is implemented in a centralized and a distributed way. The centralized version uses an area as an information sharing centre where robots reports their current status and a given a new task assignment. Oppositely, the distributed version uses local sharing of knowledge to nearby robots in order to evaluate the current status of the environment. As the distribution of the current state is done dynamically during the run time, the algorithm is adaptive to the change in workforce and to the relative demand of the different tasks.

The algorithm is observed in a simulation environment that bests re-creates a physical environment. The task allocation method is tested upon five categories that highlight the strength of a swarm robotic system: Its robustness, its reliability, its scalability, its versatility and its adaptability.

Both version have shown strong results in the conducted experiments.

* could be more of a conclusion

Experiments

The systems through a set of experiments to best describe, observe and highlight how well or how bad they perform in the five categories. Then a first experiment intends to highlight the efficiency of the DAITA system in relationship to its communication performances. A second experiment observes the task completion rate of each system and comparisons are made backed with metrics. Then, tests on the robustness of the different systems are performed. Finally, tests and comparisons on the systems' ability to adapt in the change of workforce are tested and comapred and backup with metrics.

<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

<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. ()