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

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

Payam Zahadat (90%)

Kasper Støy (10%)

Use present tense

TODO: use CAITA and DAITA and AITA for the core algorithm

TODO: say somewhere that AITA is used to define the shared part of CAITA and DAITA

TODO: all task n switch graph need to be redrawn

TODO: for each graph: it would probably be wise to always write the experimental settings as a reminder.

TODO: make sure I say “simulation step” everywhere.

TODO: should I decide and give a unit to my energy?

# Abstract

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. 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

This project also intends to precisely describe an environment and a set of experiment so that anyone can reproduce the work and compare the implemented algorithm performances with other TAs.

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

**This paragraph can also easily go somewhere else more introduction**

**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**

This project intends to explore 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.

- 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

[1]

Small chapter here that will define the set of dependent task, why it work and why it is relevant

-> 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.

<https://www.google.com/search?q=task+allocation+in+autonomous+swarm+robot&oq=task&aqs=chrome.0.69i59j69i57j69i59j69i61l3j69i65l2.832j0j7&sourceid=chrome&ie=UTF-8>

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

-> this one for instance talk about “scalability” and how it is important that the task changes in function of the needs. This is different from my task allocation system as as of now, the robot cannot quantify needs.

My robot are made to solve a set of task from A to Z without any arrogance on the speed of the execution

<https://people.idsia.ch/~frederick/taskallocation.pdf> -> this one has flying robot .. not yet relevant to me.

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

<https://www.researchgate.net/publication/2472396_ALLIANCE_An_Architecture_for_Fault_Tolerant_Multi-Robot_Cooperation>

Ant-Inspired Task Allocation Model Within a Swarm of Homogeneous Simulated Robotic-Agents

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

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

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

Heavily interesting, it says that

“This leads to a huge amount of differently designed global missions and as a result to many different solutions which are hard to compare[3]. Thus in most of the proposed methods in this area, researchers have only introduced their own methods and refrained from comparing with other methods. Our scenario also possesses different features, goals and finally distinct global foraging mission compared to previous scenarios in task allocation field”

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

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

<https://ieeexplore.ieee.org/document/8023527>

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

**Talk about the model**

* **Talk about task in general, “task allocation generalist” is a paper defining the use of specifc task from specific ants. It could be interesting to talk about it**

# Ant-Inspired Task Allocation Algorithm

This section describes the main algorithm used for the swarm of robots to achieve task allocation, and how it is integrated to the simulation. 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 for **D**istributed **A**nt-**I**nspired **T**ask **A**llocation (a distributed version of AITA).

Maybe somewhere talk about the strensght of the system envisaged by Alejandro

## Introduction

The ant-inspired task allocation algorithm used for this research has been designed in 2014 by Alejandro Cornejo et al. [link] and takes the form of a mathematical model. Furthermore, the system has not been implemented or tested in a simulation environment at the time this paper is written (Q2 - 2021), which makes it a good candidate for experiments. The algorithm is a crossover between a multitude of studies of ant’s task allocation models such as the threshold algorithm [link] or the interaction-rate model [link]. -> more here

Here say on what the model is based… “the model is based on the idea that ant can quantify…[…]…”

Say what is interesting in the algorithm over others..

## The system model

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 a function **d**(T, t) as being the **demand** for a task t at a given time T. The **demand** function can depend on any aspect of the current environment, such as weather conditions, the location of the colony (in the world),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 ant performing the task. Finally, they define **q**(T,t) as being **d**(T,t) – **w**(T,t), or; the current demand for a task minus the energy currently supplied by all ant to that task at a specific given time T (this paper will refer **w**(T,t) as being the **energy difference**).

Moreover, they define a satisfying task assignment as being one where no task is in energy deficit, that is, the task is in equilibrium. Being in equilibrium for a task means that the **energy supplied** to a task t exactly matches the **demand** of the task. (maybe that’s repetition)

TODO NDBDP:

As described in the original paper (Alejandro Cornejo et al, 2014), the energy unit can be any kind of energy (watt, joules, etc..) as long as one keeps it consistent throughout the implementation.

## Model restrictions

The previous secton 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],.

The robot will have to collect resources and 1 resources is a demand of 1. More in section ..

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 placed 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.

To conclude, this leaves the following model restrictions: a robot can carry out on a task as its ability to solve it is 1 and each time a robot finishes its task, the demand decreases by 1.

Explain the communication system of report and robot memory

## Task allocation algorithm

This section is an overview of the AITA algorithm. The complete algorithm can be found in Alejandro Cornejo et Al’s research paper.

### The binary feedback function

The algorithm is based on the worker’s ability to sense its direct and local 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 supplied to 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 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 currentT ask, 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 Resting-state.

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 values the table Q can take:” [..] 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

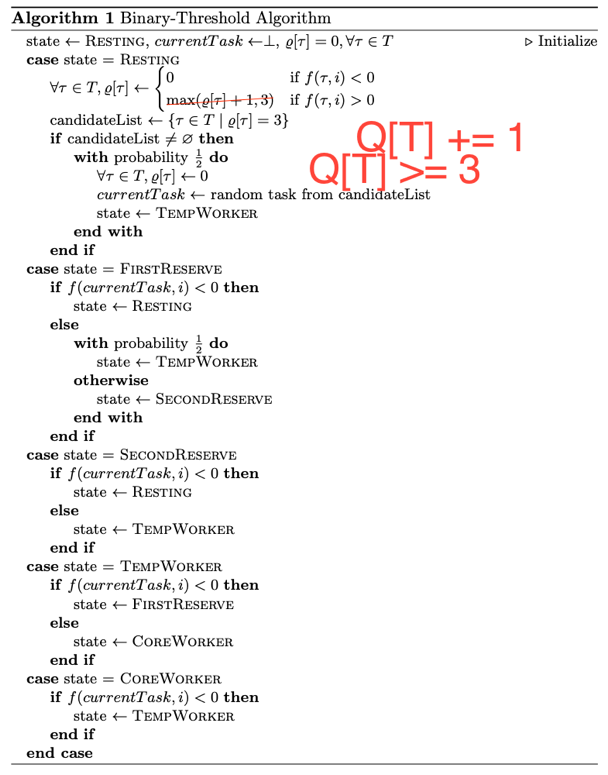
…

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 generated

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

**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**

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 and decides on a new task assignment for the robot.

Mechanisms for improving the robustness of the system against potential robots' failures have been 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 about the fact that in DAITA, a robot can receive only one information at each timestep

**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 is shared throughout the entire swarm. To do so, each worker holds an array of demands for each task 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 “advancement” 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 encountered a robot. If this elapsed time goes over a specific pre-defined simulation step, the robot is considered gone from the other robot’s system to be potentially replaced.

## Paper’s assumptions

if one task can be set to an equilibrium, then all other task will be served .. because when eq. reached, the robot are reassigned -> to be verified

- La communication failure va mettre en lumière pourquoi avoir du centralised memory est nul. Since the nest has to report to everyone, if only one fails to communicate its value there’s gonna be a big shift in the nest actual values -> not really since I have implemented a robust system. BTW talk about that in the thesis, the DAITA system that checks if information is redundant (the whole memory information updation)

# 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 system 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 another). 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 him 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 itusually perform very well in a wide range of situations, but consume a lot of computational power. Thus, this algorithm is expected to perform well but at computational costs that are high 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" (Payam Zahadat et al. [link]). Payam 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 student 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 Payam Zahadat et al.'s experiment, but the system is also expected to suffer from the consequences of applying the algorithm to the student's environment (which is discussed under the discussion section TODO).

#### **PSI Algorithm**

Each robot of the swarm holds an x value that represents their physiological age. This x value is distributed over a range of xmin and xmax (the value of the variables are defined in table n) 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 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 and GTA (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 it, please refer to the 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

A wide range of tests and experiments have proven that this 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 performance of PSI under its original setup.

##### Condition under which a worker can be attributed 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. Tests have shows that it does not impact the efficiency of PSI.

# Experiments

This section intends to present and talk about the result of the experiments. Moreover, this section precisely describes the environment and the set of assumptions and variables the system use so that anyone can reproduce it.

[intro of section]

In this section, I should also publish the base array table for PSI and all the other TA.

## Environment

All the experiments are run in a simulation built by Alexandre Vanini (this project’s author) in Python. 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. 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 following a random uniform distribution (explain why 2000 and why uniform distribution). The topology of the world is a box bound in all its directions.

Workers can carry out four kinds of tasks depending on the demand of the colony: Firstly, the idle task – which consists of resting in the nest waiting for the need to increase to a level at which the worker will be useful. 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 idle 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 that state, it is considered fully processed. Workers carrying out the idle task are shown in blue.

Moreover, each time a resource is carried out by a worker and processed, it changes 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 see the green resources and a cleaner the blue resources.

The tasks are dependent meaning that for a resource to be in the dumped type, it first has to be collected outside and brought back home. Dependence means that the division of labor is more …. This kinds of task nicely relate to real life setup such as

* Explain the task and how they could relate to real life problem ..?

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).

The own fact that I created the whole simulation with the real robot and not just pure logic is because I want to restrict the model to real life conditions to see how it does

* I am gonna let the robot finish their task before removal, because i want to be as accurate as possible.

## Assumptions and Experiments on parameters settings

In order for the experiment to be as fair as possible and to make sure each task allocation systems are given the same chance at succeeding, the environment is framed with a set of assumption an variables. Firstly, the environment assumes that all the robots are homogenous, that is, all the robots are the same, share the same capabilities and skills. Moreover, all the robots have the same navigation and robot avoidance systems. A robot is able to somewhat precisely move to a given coordinate in the plan during the simulation.

The simulated robots are Thymio-II, they so share the same abilities at sensing, moving, communicating.

which implies

* Communication time is instant, there’s no overhead -> show graph of distribution + maybe I can create a graph that shows when a robot receive a communication (at what time step) since the random system I made makes it less probable that the robot receives every step

Some variables can be changed (and applied to all TAs), these are:

* The number of food resources distributed at the start of each run
* The number of robots
* The probability of communication failure (Noise)
* The communication range
* The demand for each task
* By how much the demand for the foraging task increases each 500 simulation step

TODO maybe I should say for noise that the comm range is infinite? Also maybe see if I have to tell more about the env under which it has been tested.

TODO make sure I correctly define what the new graph of error means and depicts

TODO: For communication and noise, I can also use the graphs of robot assgigned. To show the preciseness.

What do I want to experiment?

The implemented algorithms, CAITA, DAITA, PSI, RND, and GTA, are tested in an agent-based simulation. The experiments intend to assess the efficiency of the system 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.

For this reason, we instead seek task assignments that minimize the squared difference between the energy demands and the energy supplied. Formally, an optimal task assignment is one that minimizes P τ∈T q(τ, t) 2 .

Here say that noise and range test are tests and proof for reliability and robustness, also explain that in the noise and range result’s discussion.

Noise and range could also have been tested in dynamic env to prove that they can’t adapt so much,

TODO the name of the measurement is swarm’s perception error of the actual environment, and the consequence of a high error ^is called The knowledge error shift

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

The influence of different noise levels has been tested on both version of the implementation of AITA prior to the experiments to select a bias that would be equal for every TAs. 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.

**Noise DAITA**

The first test performed on DAITA consisted of using 40 robots and make them collect 150 resources as fast as possible. The tests started with a probability of communication failure Pnoise of 0 and ended with Pnoise = 0.99. As can be seen in figure N (todo), the different levels of noise tested 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 the swarm’s perception error 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. These slight variation 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 every robot, at each simulation step, tries to share 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 packet is of; 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 39 chances over 39 to receive a packet, or a likelihood of 100%. Now, in a system where the probability of success is of 0.3 (that is, Pnoise is set to 0.7), the number of successfully sent and received packet is of; 0.3 \* 39 = 11.7. The receiver (who can still only receive one packet per round) is probabilistically speaking receiving 11.7 packet in average each round, enough for the system to update and spread the shared information globally. When the probability of success drops at 0.01, it means that the robot is receiving in average 0.39 packet each simulation step, effectively meaning that a robot will successfully receive a packet every 2.56 simulation step in. When reaching a higher probability of communication failure and a low success rateis so low that a robot does not receive a packet each simulation step, the swarm struggles to agree on the state of the current environment. This tendency of difference between the status of the environment and what is seen by the swarm can further be seen with Figure M. Figure M shows that 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 dynamic environment – Which means high oscillations in the task allocation (see figure N), resulting in a high number of task switch for each robot (see figure N). Inversely, for a high probability of communication failure, the swarm struggles to adapt quickly to the changing environment as failure in communication means information is shared at a slower pace, which results in a low oscillation in the distribution of labor (see figure N), leading to a lower number of task switch for each robot (this explains why Pnoise variate slightly in Figure N). The consequence of delay in information sharing caused by high probability in communication failure, also called *knowledge error shift*, can also be seen in the distribution of task in the set of graphs in Figure N, where, as Pnoise grows higher, the task distribution is smoother. The higher the *knowledge error shift* is, the unhealthier the system will be in the long run, ultimately meaning that though 0.99 of noise seemed to perform as good as any other level of noise, it will not be able to sustain a consistent result for long period of activation. -> is this true?

Chart, histogram

Description automatically generatedChart, line chart

Description automatically generated

|  |  |  |
| --- | --- | --- |
|  |  |  |

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 distribute 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.

TODO mettre les graphs avec 150.

**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.

TODO it could be interesting to speak about the spike as they are caracteristique of a healthy system (increase of resources, robot adapting) -> very good alex.

TODO les graphs abec 13 et 15m servent a rien.

13 metre c’est deja l’arène entire parce que l’hypothenus fait 12.qqch ..

Pour la range, je vais dire que infinite (or 13 ) is chosen as its will offer the most fair and consistant result for all the TAs

TODO maybe -> long range device for instance .. one of the paper talked about that, find it.)

### Communication -> think about how to approach what does the noise influence on.

The influence in the change of communication range has been tested before the experiments to observe and highlight the effect of it on the system. The test assumes that a communication device 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 test consisted of using DAITA with 40 robots and make it collect 150 resources as fast as possible. Figure N seems to show that ranges above 1 meter offer the most consistent result as their task completion rates are similar looking. The effect of a short-range communication distance can be correlated to the noise *knowledge error shift* in communication failure as the shorter the range is, the slower the robots are notified about changes in the environment. This effect explains why short-range communications demonstrate a slightly different curve than long-range ones. To furthermore explore the effects of the communication ranges, the swarm’s perception error of the actual environment has also been recorded and is shown in Figure N.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

As can been seen in figure N, the ranges of 0.1 and 0.5 meter offer the worst speed in environment adaptation resulting in a high *knowledge error shift* whereas range over 1 meter and included offers an acceptable level of error. This can furthermore be seen in the the number of task switch per robot shown in Figure N. System that have a slow adaptability to the environment have less task switch as the need for tasks is not shared, whereas robots with long-range communication access the information faster resulting in a higher number of task switch per robot.

A solid characteristique of the environment which is directly related to a healthy system is the visible peaks in the distribution of labor as can be seen in the series of graph in Figure N. These spikes happens as the demand increases periodically, 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 increase of demand in the foraging task, sign of a slow communication. Inversely, graph n is the labor distribution of the range ????? TODO meters, where clear sign of swarm adaptation are seen.

Make a conclusion here that since It aint a healthy system, it is bad on the long run (with 0.1 0.5 and 1 meter of range

TODO I could also use les graphs the ant assigned.

Chart

Description automatically generated

Chart

Description automatically generated

**Chart

Description automatically generated**

**What range is above lol**

**Chart, histogram

Description automatically generated**

**Chosen range**

The rest of this paper assumes that each communication is made with a range that covers the entire arena. As shown in the results of the range test, 5-7 meters would also work and be more realistic. However, the aim is to deliver every simulation and TA with the most fair and consistent setup. TODO rephrase maybe?

**Talk about the restrictions and assumptions**

**What is important:**

* Being able to denote all the variable I can act on
* Say why I would act on one more than an other
* Choosing which I want to keep the same and why (and vice-versa)

Explain how CAITA works

Thought task switch – [2]

TODO: argue somewhere that if they are capable of doing these three tasks they can effectively do ANYTHING else even if independent

TODO: Change the name with something that make sense so it reflects what is used in PSI to introduce the experiments.

Maybe that’s not the best place this could go

**Experiment**

The different task allocation systems described earlier go through a set of experiments to best describe, observe and highlight how well or how bad they perform in the five categories. A first experiment intends to highlight the efficiency of the DAITA system in relationship to its communication performances. (TODO, maybe exp about noise? like run FAITA with multiple levels and see task completion rate, if so, change the "a first experiment" and say something like " Tests on environment variable are performed ...). 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.

Each of these experiments is run 5 times and then averaged to give a more fair outcome compared to a single run. Running the experiments 5 times also enables the demonstration of the reliability of the implemented system as one can compare each run and see if the “deroulement” and outcome are approximately the same each time.

## Metrics

To rewrite

The data generated by the experiments are then processed to generate metrics. The experiment will compare the speed of execution, the total covered distance for all robots, the distribution of robots in the different classes of workers over the simulation. Moreover, the demand is tracked over the simulation and the number of robots allocated to it as well. Furthermore, each system tracks the number of switches for each individual. For the systems that have a distributed communication (DAITA, GTA, and PSI), the average sensed demand by each worker is also compared to what the actual real demand for each task is. For CAITA, the demand and the current task allocated to the workers are also compared for the real world and to what the information center sense.

Experiment on parameters settings? As in symmetry

Also with that I could use a perfect system (no noise) then experiment on the range a completion rate. Then maybe choose something that is closer to reality and redo all the experiment with that?

TODO les graphs devraient aller jusqu’a 50, refaire le plot

Todo to inclue in this paragraph: reliability and scalability

TODO changer la taille des graphs pour que ça soit cool à l’œil. Il faut rendre la thèse visuellement cool.

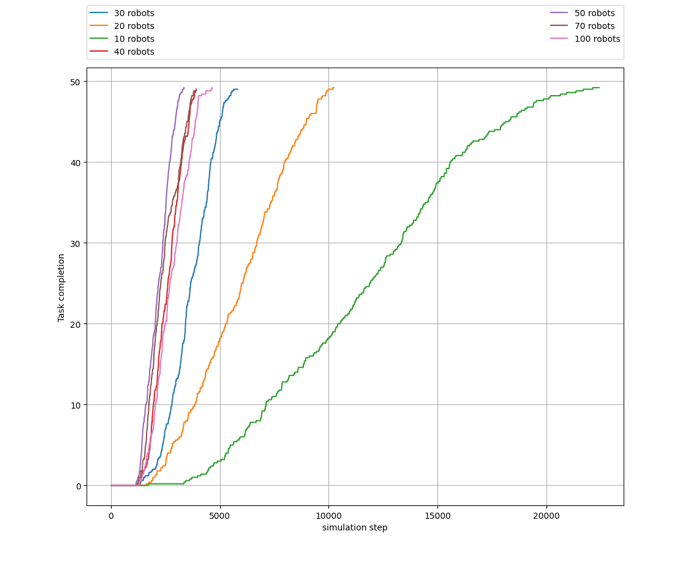
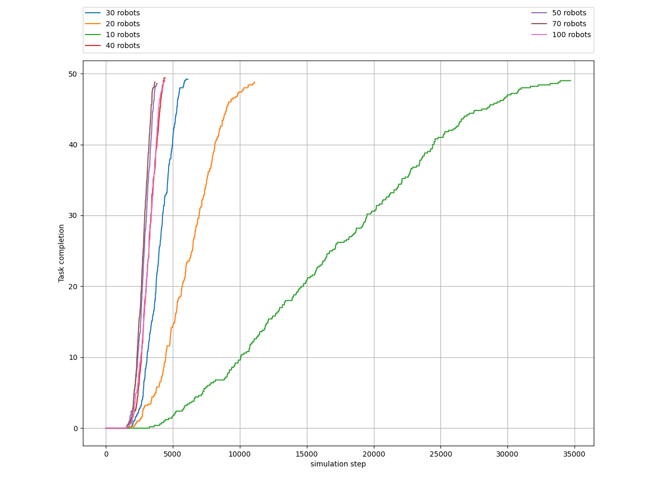
## Task completion rate

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 50 resources as quickly as possible. At each new start, the demand for the foraging task is 50 resources, and the demand for the nest processing and cleaning tasks are both set to 0. Also, the foraging demand increases by 5 for every 500 simulation steps to keep the system busy.

**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 show (figure N and N) that the 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 it is not given that within the same environment, the more robots used to complete a task, the faster the swarm reaches its goal, as both CAITA and DAITA demonstrate struggle when the number of robots goes above 50. In the former, using 70 robots is barely as good as using 50 robots, and a set of 100 robots obtains the same completion rate as 40 robots. In the latter, using 50 robots remains the overall fastest as 70 robots is as performant as 40 robots, and using 100 robots performs worse than 40 robots. A Part of this drop in performances 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 optimization of the task completion rate by improvement of the positioning of the different areas is possible, however not discussed in this paper. 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. This “underperformance” is furthermore investigated in the next section – Task completion rate on different task allocation systems.



**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. In figure (todo N)

Chart, line chart

Description automatically generated

* Results

Here, I can also show the relationship of the results compared to the solo graph

* 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

## Robustness of the system

The second experiment (EXP2) is a robustness experiment designed for the TAs to show their performances over a period of 30’000 simulation steps. The system starts with a foraging demand of 25 and both the nest processing demand and the cleaning demand set to 0 (Note de bas de page 1:). Then, the demand for the foraging task increases by 7 (instead of 5 previously) for every 500 simulation timestep. Tests have proven that using 7 over 5 keeps the system busy at its max capacity for the whole period, whereas 5 usually keeps the system in a lazy state, with more or less half of the workers busy on average, over the whole period.

* Results

## 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 optimization goal as changing variables of the environments such as the communication efficiency, the robot’s efficiency at task solving and more can influence it. 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 at a given timestep, and re-introduce it later on. The system starts with a foraging demand of 25 (Note de bas de page 1), and this time both an increase in the foraging demand of 5 and 7 is tested.

The class of workers being removed is the nest processors, [explain why]. The class is removed at the 20’000 simulation step for the system to have 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.

Note de bas de page 1: Since the CAITA and DAITA systems make use of idle states, one could think it is unfair to start the system with a foraging demand of 25 as for a simulation of 40 robots, 15 of them would be inactive at the start compared to 0 for PSI, GTA and RND (since none of them uses idle robots). However, and as will be visible in the result section, even though the system partially starts inactive, it quickly becomes busy resulting in all robots being requested for work, keeping the experiment fair and accurate.

**Adaptive change in workforce with variating number of robots**

A first set of tests is performed on the CAITA and DAITA systems with variating number of robots from 10, 20, 30, 40 to 50 here I will be able to show the shift in knowledge as robot that are suddenly removed can’t share their status

* Results

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

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

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

## Robustness of the system MORE?

To demonstrate the robustness of the system, noise in the communication has been introduced in the system, see section “noise”.

## Results

For me, robot that are not temp or core are considered as unassigned. But they are effectively assigned to a task, just waiting for the demand to raise to a level for which they would actually have to work

* **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
* Say that psi was not designed for such task, which will explain a lot of thinngngngnggs

**Discuss and experiment on lemmas and theorem of the task allocation model**

* I can also say (this paper does not intend to cover the proof established in the paper, but more to assess its efficiency though a set of experiments and comparisons to established and well tested models such as .. “

They made assumption that I need to verify and discuss

“How much robot does it take to have n task at equilibrium” “following lemma tatata .. let’s try and see” “it did not reach .. but if we do that .. the result are better .. blah blah blah”

“How much task can n robot whit stand”

*#! as of now, the task handler makes sure the robot is not assigned a new task if he carries a resource*

*#! obs: the robot are usually deposing resource in the middle but the maintenance only scan the edges (when no avoidance)*

*#! 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.*

*# ? my tweak with the >=3 fixes it*

*#! obs: sometimes an ant nest processing can lose its task assignemnt by going outside the border and be replaced by another once.*

*#! that is the same issues as descibred line 276*

*#!obs: a robot with AITA will not change task unless its task's demand is satisfied first. even if the other task has hiiigh demand.*

*#!obs seems to bring a lot of congestion since they are all trying to go at the same place*

*#! sometimes the robot will be oscilliating between task and no task, the sensor will go outside the zone*

*#! > even though the robot did not intend to leave the area, but because outside HOME, the robot keeps its task.*

*#! > it varies between has\_to\_work and not has\_to\_work so when the sensors leave the area HOME the robot does not have to report*

*#! > and will keep its state ...*

*# ? but is what I did the best option now? (go\_and\_stay\_home)*

## Discussions / Conclusion -> sort of

**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

# 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 ..

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