ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Seconda Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

DIVISION OF LABOUR IN A GROUP OF AGENTS INSPIRED BY ANTS' FORAGING BEHAVIOUR

Elaborata nel corso di: Fondamenti di informatica LB

Tesi di Laurea di:
PATRICK PITTOLA

Relatore:
Prof. ANDREA ROLI

[illegible]

ANNO ACCADEMICO 2010–2011
SESSIONE III

KEYWORDS

Swarm intelligence

Division of labour

Adaptive behaviour

Contents

Introduction

One of the issues in computer science is to design artifacts that simplify the everyday life. The evolution of technological requirements often requires the evolution of both hardware and software solutions.
Unfortunately, sometimes classic ‘divide et impera’ approaches of computer science might not be effective or suitable, so new approaches are sought. Several non classical techniques were proposed to overcome the limitations of classical ones, most of them inspired by Nature.
The technic I will illustrate in this thesis is inspired by natural “Swarm Intelligence”(SI).
SI has its origin in biocomputing, as it takes inspiration by natural systems able to solve non trivial problems. This field uses as a metaphor the behaviour of animal species, especially social insects, to create new algorithms and solutions.
In my thesis I had developed a simulator of a natural phenomenon, useful to the biologist to verify ants behaviour in nature and for computer-scientist to get in touch with this problem solving technique.
The example I used to illustrate the characteristics of SI is inspired by the work of Labella (2006). I developed a software application that simulates the behaviour of a colony of ants performing a foraging task.
The focus of this work is on ants division of labour; indeed, this kind of system is really flexible and performant.

The whole application was written in Java and developed with Eclipse.

1. The first chapter is an introduction to Swarm Intelligence. It starts with a brief section summarising the foundation and history of this field; than it introduces the main proprieties of this kind of system and explains how it could be useful in real-world applications. Finally, focus moves to some examples in both biology and computer science.
2. The second chapter focuses on the specific application, starting with problem description and system requirements, defining goals, constraints and the reasons for all the choices made during this design process.
3. The third chapter describes the structure of the software application, illustrating classes and objects in the developed system, their behaviour, functionalities and

interactions with the rest of the system.

4. The forth chapter shows experiments and tests used to verify the system functionalities.
5. The fifth concludes the thesis. I summarise the goals achieved and I show several possible future improvement of the present application.

CHAPTER 1
Swarm Intelligence

In this chapter i will explain what the field of “Swarm Intelligence“ is, I will describe his characteristics and i will illustrate how this studies could be useful, also giving few examples

1.1 Introduction to Swarm intelligence

As mentioned in the introduction, the scientific field of Swarm Intelligence was originated from biology studies on social insects behaviour, which show an incredible ability to solve complex problems; indeed, they can solve some tasks that largely exceed the capabilities of the single insect, through an unusual coordination patterns.
Social insects behaviour fascinates biologists and naturalists since decades; in the past, they even speculated about the existence of a super-agent coordinating the colony, such as the queen ant.
For some species the behaviour seems very surprising, like e.g., for ants and termites, which apparently seem to have a good representation of their huge and mazy nests, building and crossing enormous underground structures.
Recent studies have controverted the previously hypothesis; in fact, these insects lack of an internal hierarchy or a supervisor who coordinates the nest activities. The queen has not duties as supervisor or coordinator and there is nothing like an invisible presence or a global conscience. Moreover there is not an inter-mate coordination or direct cooperation an the individuals have not a global representation of the nest or of the environment. As proven by some biologists, these animals do not need a direct coordination or a global knowledge of the environment or of the nest; they can perform in reliable way a lot of work using only local information. Indeed, they use a special mechanism call “stigmergy”, which is one of main properties for this behavioural pattern.
The term “stigmergy” was introduced by Grassé [1959] and it was originally used to describe the behaviour of two termites’ species observing nest’s building. It was aimed to describe a sort of indirect communication mediated by modifications of the

environment. This mechanism is also quite used by other animals species. Grassé in his study showed that the coordination in building activity does not directly depend on the workers, but it is mainly guided by the rising nest structure. Indeed, every termite uses only local information to build the structure: every time one of the workers modifies the structure, the new configuration influences the action of the other insects who could decide to add building material to the same point or start a pillar in another location. This process leads to an almost perfect coordination of the collective work and may give us the impression that the colony is following a well-defined plan while the working coordination passes only through environmental coordination and insects interactions with it.

1.2 Swarm Intelligence in computer-science

Several of the classifications use in this paragrapher are inspired by Thiemo Krink, “Swarm Intelligence Introduction“..
The need of new type of patterns and algorithms comes from recent aims and evolution of computer-science. In facts, traditional software techniques have several drawbacks because they have to be:

- well-defined
- fairly predictable
- computable in reasonable time with serial computers

For several applications, these limitations are to strong, and do not suite with certain class of problems with certain characteristics like:

- well-defined but computational hard problems like “Travelling Salesman Problem”
- hardly predictable problems and dynamic ones like real world autonomous robots or management and business planning

To solve this kind of problems was elaborated new solutions based on different computing style like:

- DNA computing
- Quantum computing
- Bio-computing

Swarm Intelligence could suite very well as way to develop problems solution.

Obviously this solution could have some drawback too, especially because natural systems sometimes could fail or even if successful could waste some resources. While in biology the failure of a single agent could not dramatically influence the system’s performance, this could be true for human applications.
Another limitation could be the compromises made in nature between several factor, not always admit in other context. Even the not completely understanding of natural phenomenas could be a drawback for this technic.

Despite the drawbacks of SI, this kind of systems find several applications like:
- Swarm robotics(approach consisting in coordination of multi-robot systems, made by several simple physical robots);
- Discrete optimization (using ant algorithms for example) ;
- Graph partitioning;
- Collective decision making.

For all these correlation between the two science fields of biology and computer-science, the study of both discipline could help the other. Biology could give new metaphors and inspiration to computer-scientist, while the computer simulation of biological events could help the researcher for that field of study.

1.3 Mechanism and characteristics

Ants are particularly interesting insects in this context, in fact:

- ants solve complex tasks by simple local means;
- ant productivity is better than the sum of the single activity;
- ants are “grand masters” in search and exploring.

The principals mechanism of natural organization are self-organization and stigmergy.

1.3.1 Self-organization

Self-organization is a set of dynamical mechanisms whereby structures appear at the global level of a system from interactions of its lower-level components.
From Bonabeau (1999)
It relies on four basic ingredients:

1. Positive feedback
   Some of stimulus that create new stimulus
2. Negative feedback
   That ingredient counter balances positive feedback and leads the stabilization of the collective pattern.
3. Amplification of fluctuations
   A lots of action performed by insects could by describe as stochastic, this propriety is really interesting because work well in comparison with some computer-science application and because this random fluctuations enable the colony to find new solutions.
4. Multiple interactions and stigmercy interactions
   Required among individuals to produce apparently deterministic outcomes and the appearance of large and enduring structures.

In addition to the previously detailed ingredients, self-organization is also characterized by a few key properties, in fact self-organized systems:

- Are dynamic: Thanks to the ingredient listed above, the ants are able to react quickly to environmental variation leading to a flexible model;
- Exhibit emergent proprieties: Thanks to the nonlinear combination of the interactions between the members of the colony, this kind of systems display properties that are

more complex than the simple contribution the single agent;

- Often lead to bifurcations: A bifurcation is the appearance of new stable solutions when some of the system's parameters change This corresponds to a qualitative change in the collective behaviour;
- Can be multi-stable: Multi-stability means that, for a given set of parameters, the system can reach different stable states depending on the initial conditions and on the random fluctuation

1.3.2 Stigmergy.

The original definition of stigmergy was: "Stimulation of workers by the performance they have achieved". In this context is used to define a form of indirect communication mediated by modifications of the environment

1. Indirect agent interaction modification of the environment
2. Environmental modification serves as external memory
3. Work can be continued by any individual
4. The same, simple, behavioural rules can create different designs according to the environmental state

Good examples of stigmergy in nature are the nest building activities in wasp and termites.

1.4 Example of Swarm Intelligence in nature

There are several useful examples extensively studied in literature to show how social insects act and which kind of contribute they could give to computer-science.
I will three examples, the first two examples talk about ants, because they are the species more studied in both biology and computer-science field. The third example talk about wasps.
The examples I will discuss are:
1. Prey retrieving in ants' colony
2. Division of labour in ants' colony
3. Nest building by wasps

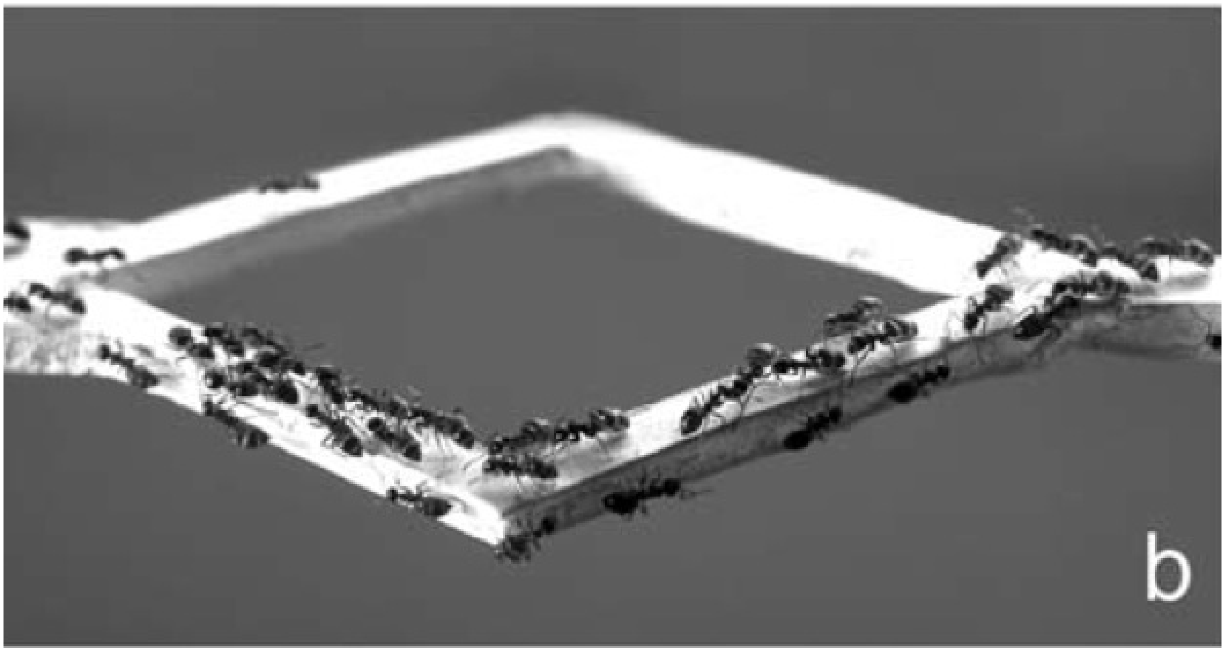
1.4.1 Prey retrieving

To perform this task the ants go out from the nest and move randomly in the environment until they find a food source. If an ant finds a food source, it comes back to the nest releasing a pheromone, if other ants smell that pheromone, trail they will cover that path, retrieving food source and bring it to the nest. Every ant which cover that path incremen the trail's intensity. This implement the mechanism of positive feedback.
That trail slowly evaporate. If the ants do not cover the trail, the trace evaporate completely and the ants do not continue on that path. This implement the mechanisms of negative feedback
Thanks to the combination of this two effects, the ants usually find the shortest path to reach the food source.
Deneubourg (1990) showed how the ants follow that behaviours in a series of experiments.
A food source is connected to an ant nest by a bridge with two equally long branches. When the experiment starts the ants select randomly and with equal probability, one of the branches. Because of statistical fluctuations one of the two branches is chosen by a few more ants than the other and therefore is marked by a slightly higher amount of pheromone. The greater amount of pheromone on this branch stimulates more ants to choose it, and so on. This autocatalytic process leads very soon the ant colony to converge towards the use of only one of the two branches.

Using two branches of different length, the first ants coming back to the nest are those that took the shortest path twice (to go from the nest to the source and to return to the nest), so that more pheromone is present on the short branch than on the long branch immediately after these ants have returned, stimulating nest mates to choose the shortest branch..

This capacity to find the shortest path result really useful for a lot of real world applications and to solve several problems well know in scientific literature like:

- Travelling Salesman Problem: find a closed tour of minimal length connecting “n” given cities;
- Scheduling problems: the process of deciding how to commit resources between a variety of possible tasks;
- Vehicles routing;
- Routing on computer network.



CNRS Photothèque Gilles Vidal and Guy Theraulaz

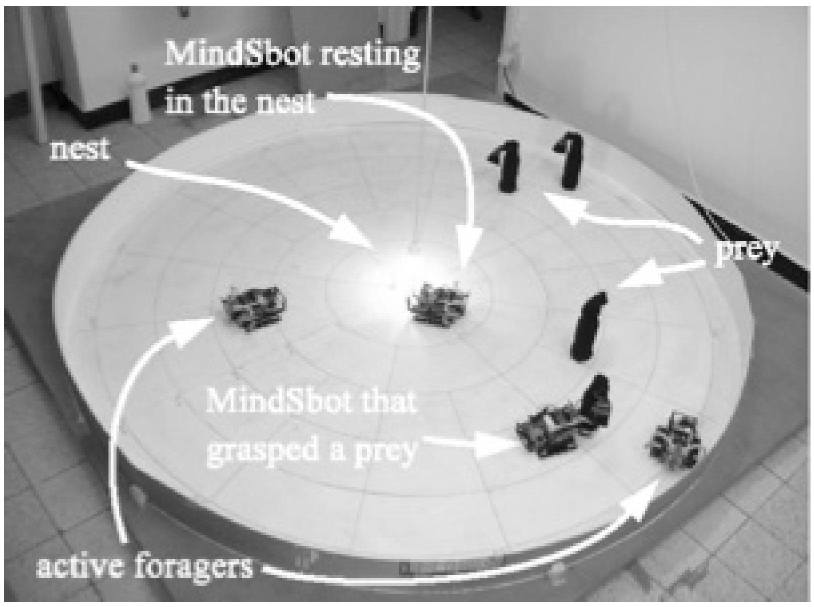
1.4.2 Division of labour

This is a main feature into the social insects colony.
Several factor influence the division of labour into an ants colony like:

- Caste;
- Morphological factors: workers that belong to different morphological castes tend to perform different tasks;
- Age: age subcastes correspond to individuals of the same age that tend to perform identical tasks;
- Stimulus: the colony's worker could change the executing task in respond to stimulus coming from inside or outside the colony.

The last factor is the key for the ants' flexibility, in fact the ratio of individuals busy in a certain work could change in respond of various stimulus and without direct communication.
Bonabeau (1996) has developed a simple model for task allocation in ants, based on the notion of response threshold: individuals start to become engaged in task performance when the level of the task-associated stimuli, which plays the role of stigmergic variable, exceeds their threshold.
Differences in response thresholds may either reflect actual differences in behavioural responses, or differences in the way task-related stimuli are perceived. When specialized individuals performing a given task are withdrawn (they have low response thresholds with respect to stimuli related to this task), the associated task demand increases and so does the intensity of the stimulus, until it eventually reaches the higher characteristic response thresholds of the remaining individuals that are not initially specialized into that task; the in-crease of stimulus intensity beyond threshold has the effect of stimulating these individuals into performing the task.Dorigo (2000).
Differently, non specialized worker have higher thresholds and need higher stimulus to be induce to perform the incoming task and use to continue with the old task for low stimulus.
Worker who belong to different castes start with different thresholds for different works.
This model can effectively explain the ant's behaviour and their flexibility for division of labour
This example is really suitable for several problems of task allocation and to manage the agent activities in software applications and swarm robotics.

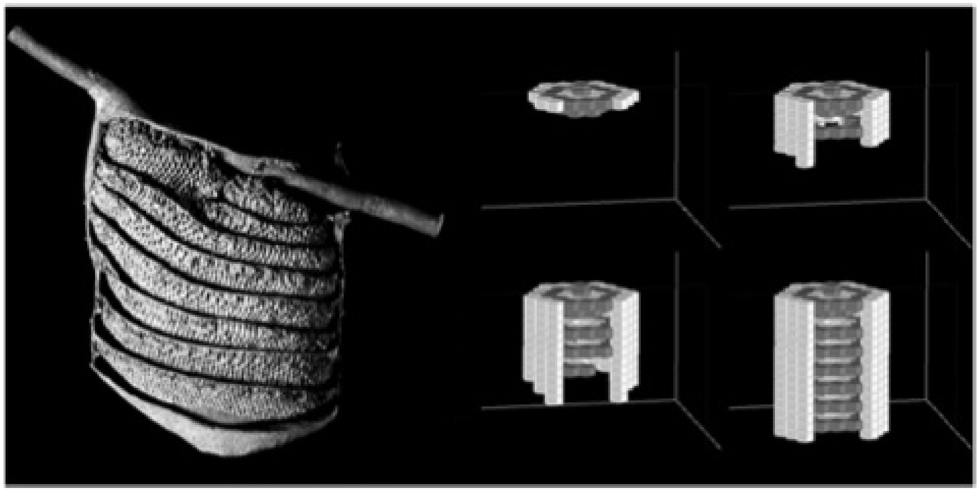
Several experiments was done using similar model with a thresholds parameter, like the one of Krieger and Billeter (2000) where a group of robot was design to perform a prey retrieval task catching pucks spread in the environment are taken back by the robots to the "nest" where they are dropped in a basket. Another significant experiment was designed from Labella (2006), where different groups of robots with different characteristics was busy into a similar task of prey retrieving.
These examples show how the experiments developed using Swarm Intelligence paradigm, could be competitive into real world application in comparison with other algorithm and methods used in scientific literature.



Snapshot of an experiments with four MindSbot busy in prey retrieving by Labella

1.4.3 Nest building

Using the observation of Grassé (1959) about termites behaviours in nest building generate the interest for model base on similar activities. One example is the work of Karsai and Theraulaz, who has assembly a model of self assembly, inspired by the nest building behaviour of wasps.
That model based on wasps activity is really interesting to explaining the concept of stigmergy. To decide where to build a new cell, wasps use the information provided by the local arrangement of cells on the outer circumference of the comb. Potential building sites on the comb do not have the same probability to be chosen by wasps when they start to build a new cell, but wasps have a greater probability to add new cells to a corner area where three adjacent walls are already present, while the probability to start a new row, by adding a cell on the side of an existing row, is very low as said by Camazine (2001).
From this observation, several model to simulate this behaviour to create self-assemble model was build, like the experiments of Theraulaz and Bonabeau (1995a) (1995b). In these works a set of agents, acting as wasps was busy in a building activity into a three-dimensional discrete hexagonal space.
Nest architectures obtained by simulations shown that the complexity of the structures that are built by social insects does not require sophisticated individual behavioral rules.



Wasp nests building from Theraulaz and Bonabeau

CHAPTER 2

A Java simulator: System Requirements

In this chapter I will illustrate the nature of the analized problem and developed project's requirements.

The case I studied and developed is inspired by Labella(2006)'s work and deal with the division of labour inspired by ant foraging behaviour.
Even if the example is a case of prey retrieving, the focus isn't on the classic mechanism used by ants to find food and bring it to the nest, but on the division of labour and behaviour changing in response to environmental factors.
The problem could be describe as Detrain (1997) and Hölldobler (1990)

> *"Ants randomly explore the environment until one of them finds a prey. If the prey is not too heavy, the ant tries to pull it to the nest; otherwise, it tries to cut it or to use short or long range recruitment (by spreading some chemical substance in the environment or going back to the nest while leaving a pheromone trail). The prey is pulled straight to the nest (pushing is never observed), both in the case of individual or collective retrieval.*
> *After the retrieval, the ant returns directly to the location where it found the prey.*
> *If, after a certain amount of time, an ant come back to the nest has a certain probability to return outside the nest to search another prey or remain into the nest to absolve other work, following various stimulus depending by the environment and the nest activity. Different ants have different capabilities and different speeds that make some agents better than others"*

Now i will highlight the most relevant words and underline most meaningful sentences that describe ant behaviour, in order to write exhaustive system requirements:

> *"<u>Ants</u> <u>randomly explore</u> the <u>environment</u> until one of them finds a <u>prey</u>. If the prey is not too heavy, the ant tries to pull it to the nest; otherwise, it*

tries to cut it or to use short or long range recruitment (by spreading some chemical substance in the environment or going back to the nest while leaving a pheromone trail). The prey is pulled straight to the nest (pushing is never observed), both in the case of individual or collective retrieval.After the retrieval, the ant returns directly to the location where it found the prey.
If, after a certain amount of time, an ant come back to the nest has a certain probability to return outside the nest to search another prey or remain into the nest to absolve other work, following various stimulus depending by the environment and the nest activity. Different ants have different capabilities and different speeds that make some agents better than others"

2.1 Glossary

Explain the word meaning and use a specific language suited for the application context is necessary to have a correct development for the project.

Name	**Synonymous**	**Meaning**
Ant	Agent, robot	A single unit who could perform the task
Environment	Arena	Real system in natural phenomena and virtual system use for the application
Prey	object	What to retrieve
Chemical substance	Pheromone	Chemical substance use to recruit other ants
Nest		Spot from which the agent start the research and where they bring the prey

Retrieving	Foraging	Searching a prey into the arena and bring it to the nest
Amount of time	Time- out	Time for an Ant to stay out from the nest before she has to came back to it
Different Speeds	Different capabilities	Space covered in a time unit

2.1.1 Relevant behaviours

Behaviour	**Synonymous**	**Relevance**
Randomly explore	Move randomly	Reveal the stochatic nature of the task
returns directly to the location where it found the prey	Move to the last prey	Relevant behavior after the capture
come back to the nest		Show the nest as starting and ending point of every search
search another prey or		Reveal the continuous nature of the problem
remain into the nest		Highlight the propriety of specialitation and division of labour

2.2 Goals

- Simulate the behaviour of real-word ants executing a prey retrieval task
- Prove the model's flexibility, especially through the following indicators:
    - Faster ants usually catch more preys;
    - The ants who catch more preys became specialist for preys retrieving task, hence they spend most of their time outside the nest searching for food source;
    - Vice-versa, the ants who catch less preys usually spend more time in the nest, dealing with other works;
    - An environment with more prey leads to a major number of prey catch;
    - The number of ants busy in foraging is proportional to the number of prey into the environment.

2.3 Requirements

In order to make a significant model, I have advanced some hypothesis. All of these conjectures are acceptable in comparison with studied natural phenomena and the application's goals and domain.

2.3.1 Bounds

Few limitations to the representation of real objects in the application were required, in order to simplify a few aspects of the project and put major effort on the relevant aspects of the problem. Therefore I applied the following bounds to the application :

- I did not consider the pheromone trail, this argument was fully developed during other study and works;
- I considered only prey that could be retrieved by a single ant, in this way I could consider preys as simple objects;
- The environment has rectangular or square shape, that leads to a simple implementation; it also has limited dimensions, that means the presence of edge on the environment's limit, an ant coul not walk over the arena's edge;
- For every time unit, only one object could occupy a certain space in the environment, with the only exception of an ant with a captured prey. Two ants could not be

contemporarily in the same space.

2.3.2 System's parameters:

In order to have a wider comprehension of the application's domain, is useful to understand which are the necessary parameters to insert in the application.
These are the general parameters set by user that aren't part of the singles entities are:

- Number of ant
- Preys re spawn.
  For statistic reasons, it is reasonable to assume that, when an ant catch a prey, this prey will be replace by another one.
  With this option disable, the captured preys wouldn't be replaced and the environment will be soon run out of prey. This option have drastic effect regarding the specialisation, in fact if the environment will run out of preys, the ants will mainly remain into the nest for an absence of prey. For the normal simulations the "prey re spawn" option will be set on.

2.3.3 Ant

From the previous description we could assume the following parameters for every ant:

- <u>Speed</u>: amount of space cover for time unit
- <u>Starting probability</u>: probability to exit from the nest at the beginning of the experiment, represent the starting response to the stimulus

2.3.4 Prey

The preys are passive entities set randomly in the environment.

2.3.5 Environment

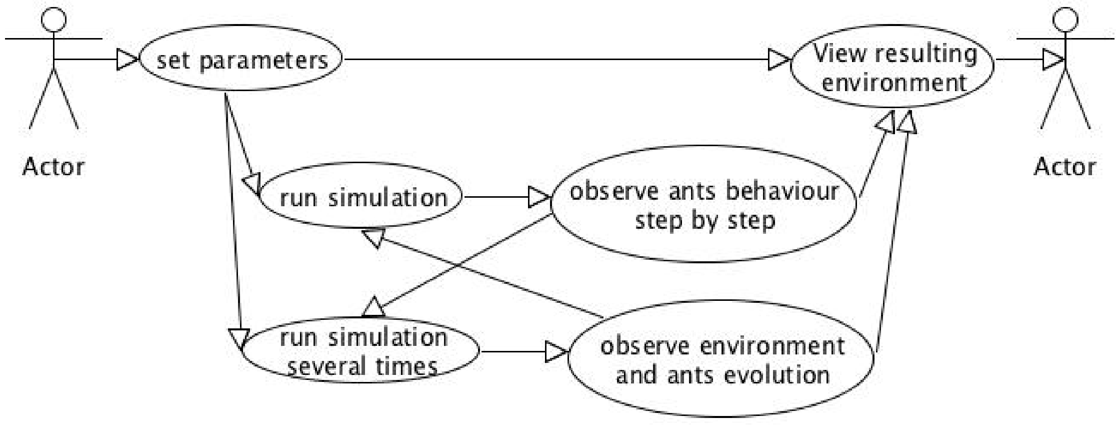
The main characteristic for the environment are:

- Dimensions: high and width of the selected environment;
- Nest coordinate: the nest is set in the middle of the arena, so the movement of the ant will be more realistic.

2.4 Use cases

Due the nature of this application, reserve to specific issue, there are only few notable use case:

- Create the system: The user take the first approach with the system, set the environment and see the involved entities;
- Run simulation once: Watch the execution of a single step do by the ant, in that way the user could easily and fully understand and verify how the various entities work and interact;
- Run simulation more times: Useful when the user has already understood and verify the ants behaviour to observe the evolution of the system during the required amount of time.


Actor
set parameters
View resulting environment
Actor
run simulation
observe ants behaviour step by step
run simulation several times
observe environment and ants evolution

2.5 Control

The ants follow the behaviour shown in the picture below:

- Rest: The ant lays in the nest waiting to exit, meanwhile it does other works different from prey retrieving;
- Search: The ant enters into this state with a probability P (better explained in paragraph 2.5.2) proportional to the number of success the animal has archived and inversely proportional to the failures. A success is archived when an ant comes back to the nest with a prey, otherwise it archives a failure. While the ant is in "Search" state, it will move randomly, avoiding obstacles, until it finds a prey or the time-out expires;
- Return: When the time-out is expired without that a prey was found by the ant, it will come back to the nest through the shortest possible path choosed to avoid obstacles. Due to the fact that the ant hasn't caught a prey, it will archive a failure.
- Retrieve: The ant has caught a prey, so it comes back to the nest avoiding obstacles and using the shortest possible path;
- Deposit: The ant arrives in the nest after it has caught a prey; it leaves that prey to the nest and increases the successes number. Then the ant goes back to the location where it found the prey, using the shortest possible path and avoiding all obstacles.

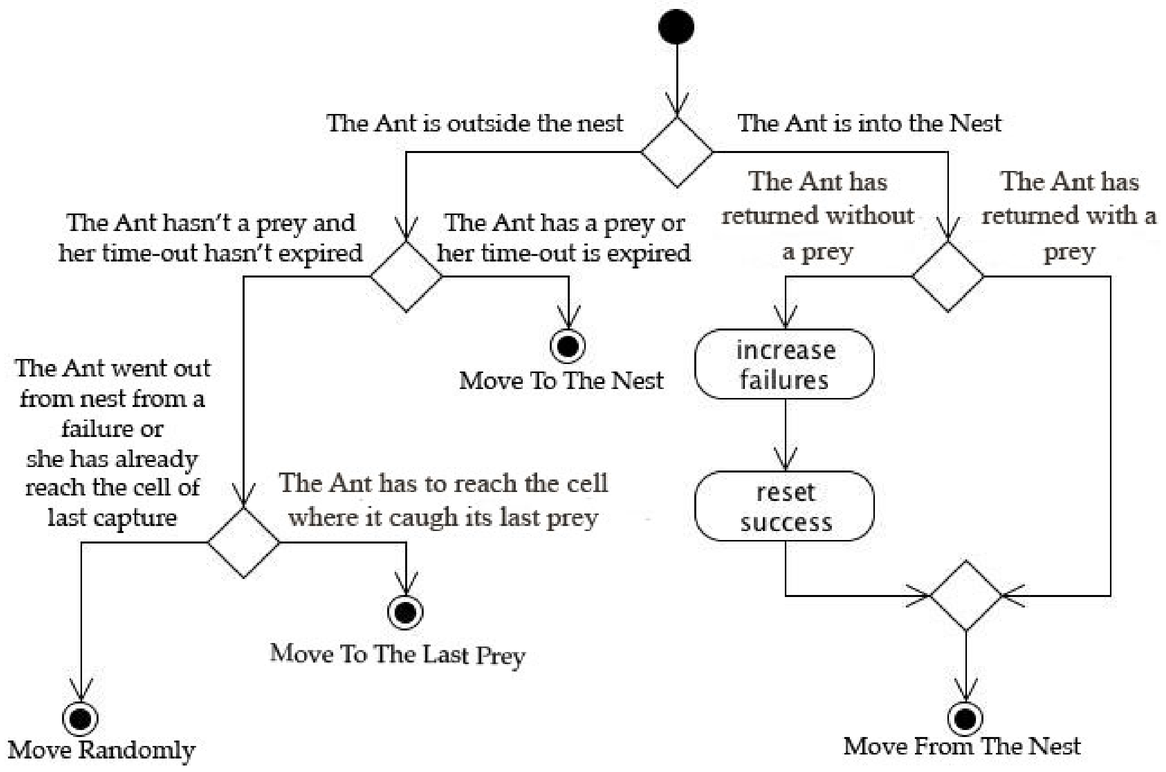
From the description above, we could deduce 4 types of movement available for an ant:

- Movement from the nest into the environment after a failure. The ant has a certain probability to exit the nest or remain inside doing other tasks;
- Random movement into the environment searching a prey;
- Movement toward the nest covering the shortest possible path and avoiding obstacles after the term of time-out or after the capture of a prey
- Movement from the nest to the location where the last prey was situated, using the shortest possible path and avoiding obstacles.

Let's describe all these movements accurately:

2.5.1 Ant's movements

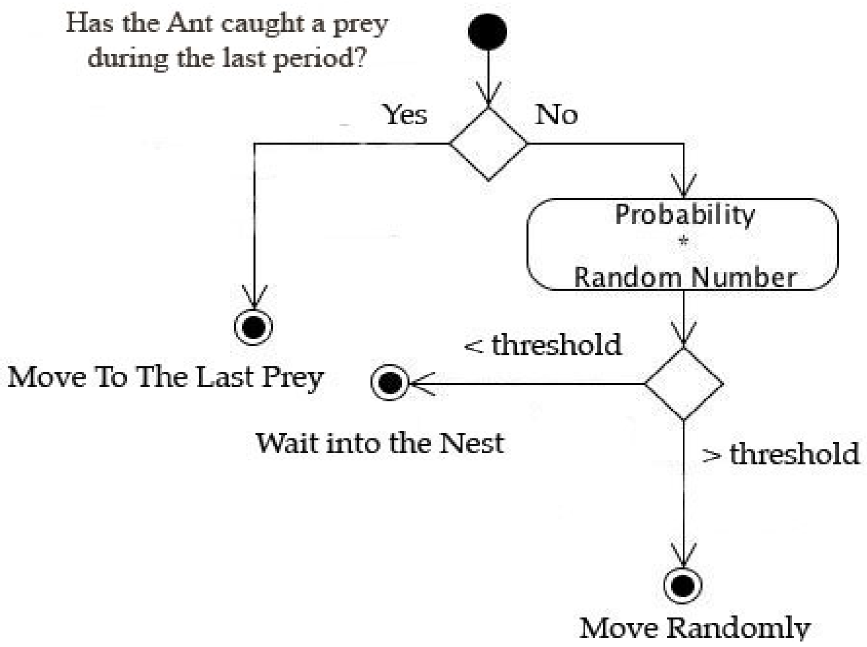
This is the schematic description for the ants' possible movements.
At the start of every time unit, an ant responds to the behaviour showed above and execute one of the action listed at the end of the selected branch.


The Ant is outside the nest
The Ant is into the Nest
The Ant hasn't a prey and her time-out hasn't expired
The Ant has a prey or her time-out is expired
The Ant has returned without a prey
The Ant has returned with a prey
increase failures
Move To The Nest
The Ant went out from nest from a failure or she has already reach the cell of last capture
The Ant has to reach the cell where it caugh its last prey
reset success
Move To The Last Prey
Move Randomly
Move From The Nest

2.5.2 Movement from the nest

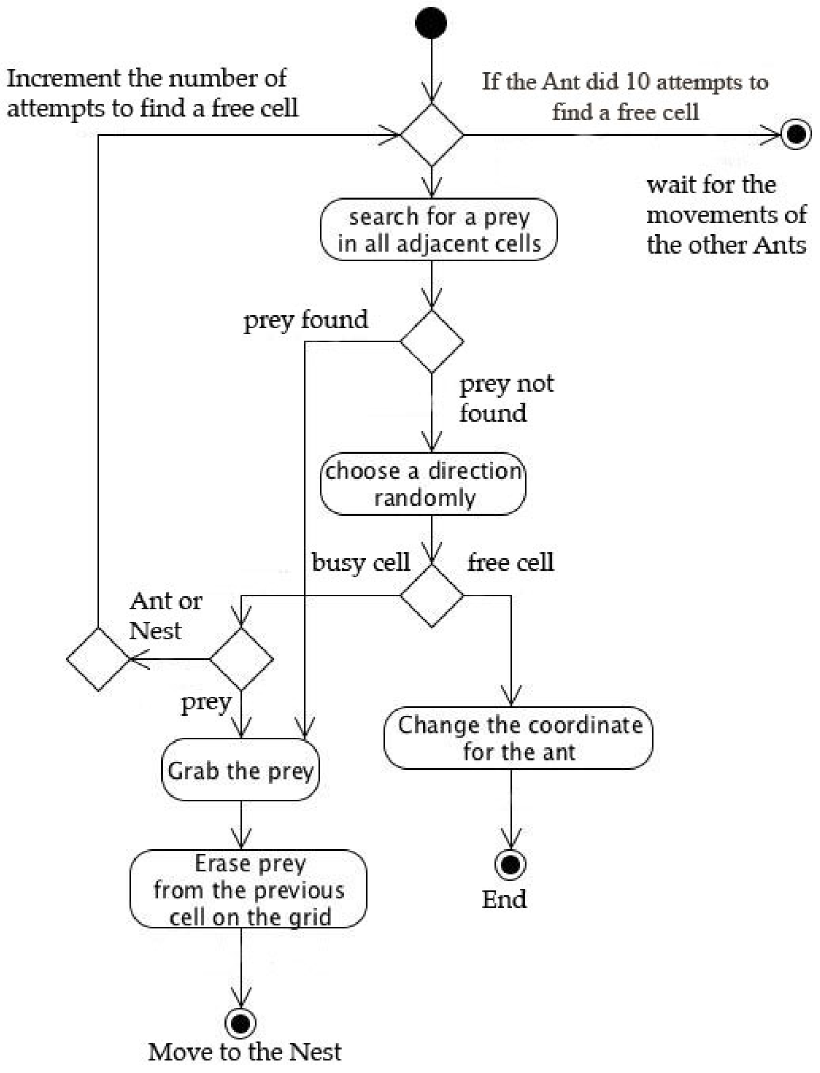
In order to exit the nest, the ant compares the result obtained by the product of a random number with the current ant's probability. If the result is higher than a pre-calculated threshold (proportional to the max successes number), the ant will exit from the nest and move into the environment for the current turn, otherwise she will wait the next turn to calculate a new product to compare with threshold and repeat the operation.
The probability is upgraded in the following way every time an ant archives a success or a failure:

if success then
succ -> succ+1
fail -> 0
P-> min{Pmax,P + succ·Δ }
	if failure then
	succ -> 0
	fail -> fail+1
	P -> max{Pmin, P + fail·Δ }
		Pmin = Minimum probability admit
		Pmax = Maximum probability admit
		Δ = basic unit for increment or decrement



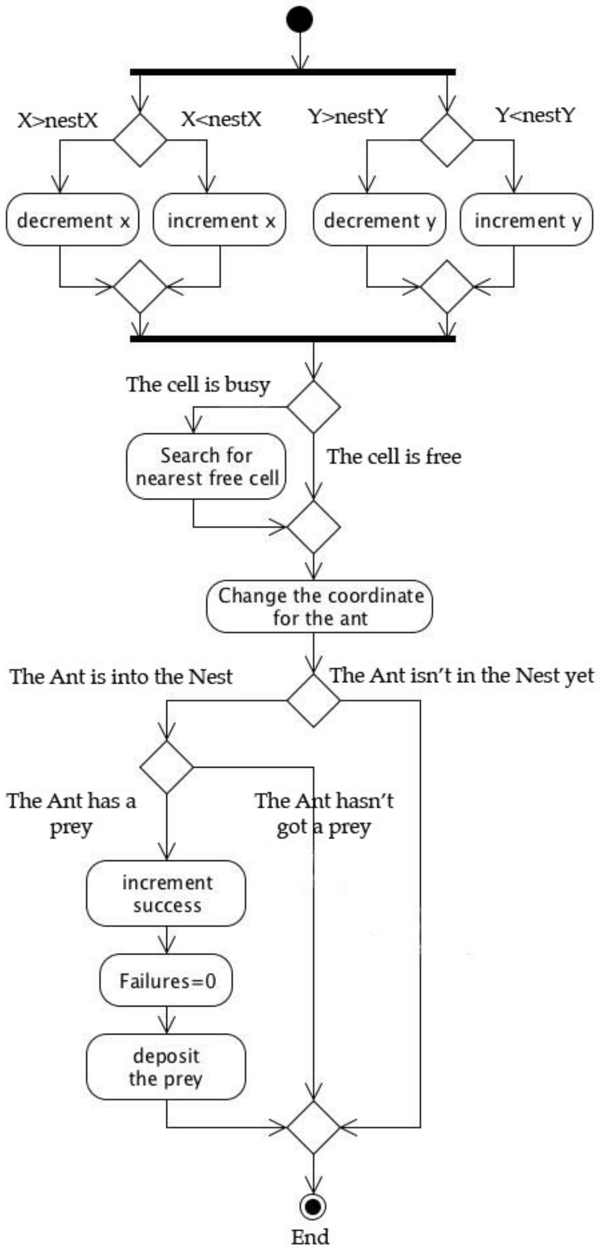
2.5.3 Move Randomly

At the beginning of her movement the ant "watches" for a prey in all adjacent cells. If it finds a prey, the ant catches it and goes back to the nest, otherwise it tries to move into one of the other cells choosing randomly. If that cell is free the ant moves there, otherwise it repeats the process until it finds a free space. If this process has been repeated for ten times, the ant will temporary remain in the current space and wait for the mates' movements, than it will repeat the aforementioned process until it finds a free cell. The procedure described in the last sentence will result necessary when an ant has not free cells to choose around her, rare event that could occur for experiments with high number of ants number and prey density.


Increment the number of attempts to find a free cell
If the Ant did 10 attempts to find a free cell
wait for the movements of the other Ants
search for a prey in all adjacent cells
prey found
prey not found
choose a direction randomly
busy cell
free cell
Ant or Nest
prey
Grab the prey
Change the coordinate for the ant
Erase prey from the previous cell on the grid
End
Move to the Nest

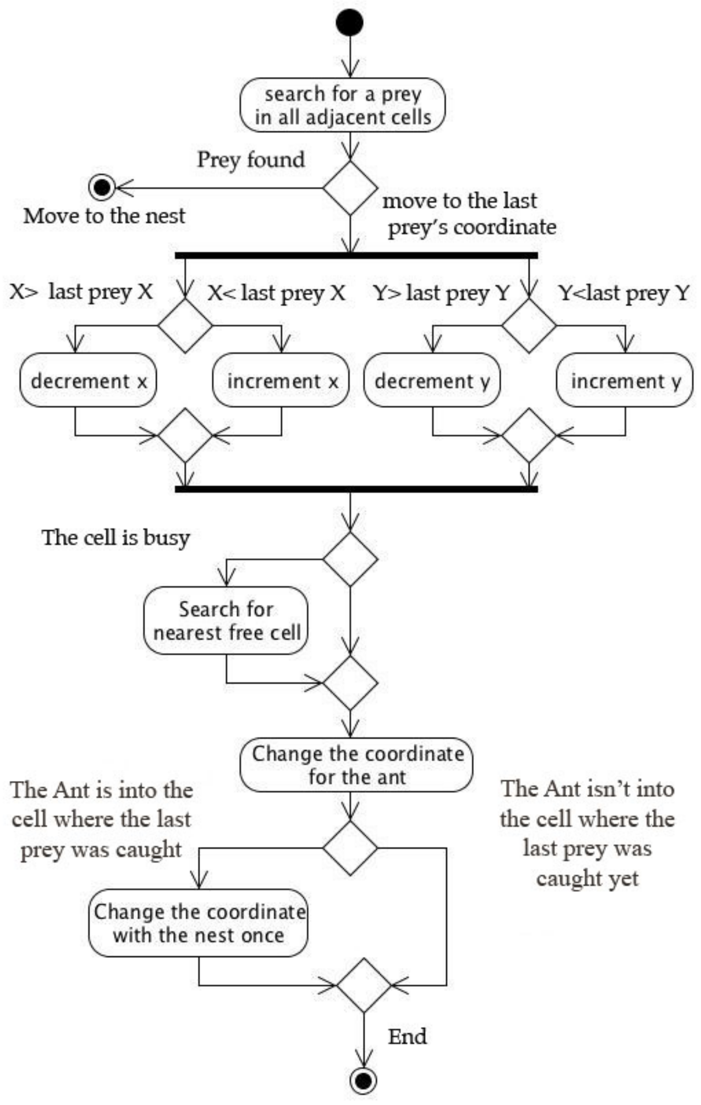
2.5.4 Move to the nest

This simple algorithm is used by ants to reach the nest using the shortest path. This procedure is necessary when the ant finds a prey or its time-out is expired. If one of the desired cell is busy, the ant will move into another sub-optimal cell, depending on the current position.
When the ant arrives into the nest, it will deposit the prey, achieve a success and increment the probability as explained previously.


X>nestX
X<nestX
Y>nestY
Y<nestY
decrement x
increment x
decrement y
increment y
The cell is busy
Search for nearest free cell
The cell is free
Change the coordinate for the ant
The Ant is into the Nest
The Ant isn't in the Nest yet
The Ant has a prey
The Ant hasn't got a prey
increment success
Failures=0
deposit the prey
End

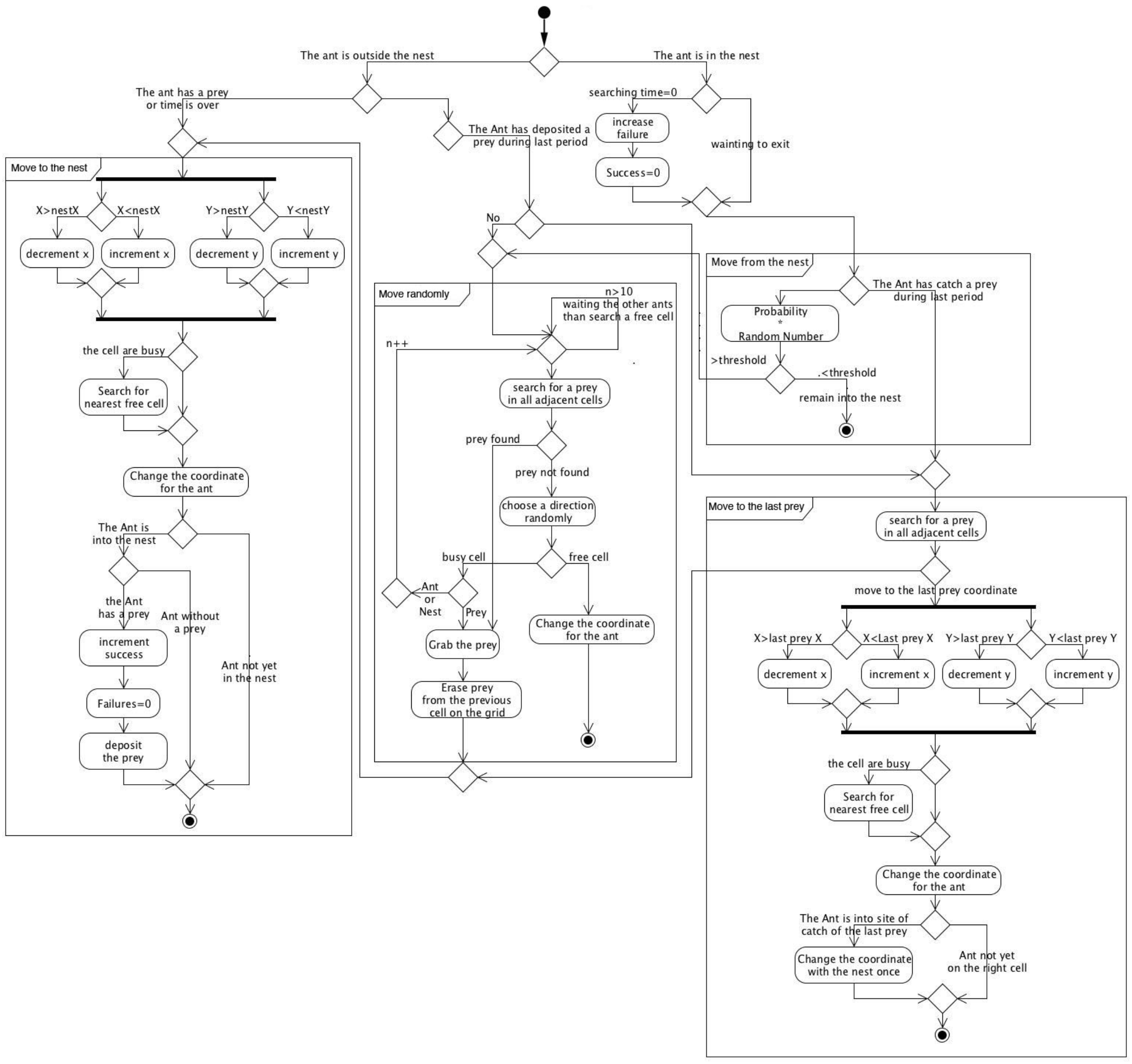
2.5.5 Move to the last prey

The first part of this algorithm is similar to the previous one, but before the movement the ant searches for a prey in adjacent cells. If it finds a prey, the ant will come back to the nest.
If the ant arrives to the desired location without finding a prey, it will start the normal "Search" state, searching a prey to retrieve..


search for a prey in all adjacent cells
Prey found
Move to the nest
move to the last prey's coordinate
X> last prey X
X< last prey X
Y> last prey Y
Y<last prey Y
decrement x
increment x
decrement y
increment y
The cell is busy
Search for nearest free cell
Change the coordinate for the ant
The Ant is into the cell where the last prey was caught
The Ant isn't into the cell where the last prey was caught yet
Change the coordinate with the nest once
End

2.5.6 Summary

This is the complete representation for the description of the ants' movements into the environment



The ant is outside the nest
The ant is in the nest
The ant has a prey or time is over
searching time=0
increase failure
Success=0
wainting to exit
The Ant has deposited a prey during last period
No
Move to the nest
X>nestX
X<nestX
Y>nestY
Y<nestY
decrement x
increment x
decrement y
increment y
the cell are busy
Search for nearest free cell
Change the coordinate for the ant
The Ant is into the nest
the Ant has a prey
Ant without a prey
increment success
Failures=0
deposit the prey
Ant not yet in the nest
Move randomly
n>10 waiting the other ants than search a free cell
n++
search for a prey in all adjacent cells
prey found
prey not found
choose a direction randomly
busy cell
free cell
Ant or Nest
Prey
Grab the prey
Erase prey from the previous cell on the grid
Change the coordinate for the ant
Move from the nest
The Ant has catch a prey during last period
Probability * Random Number
>threshold
.<threshold
remain into the nest
Move to the last prey
search for a prey in all adjacent cells
move to the last prey coordinate
X>last prey X
X<Last prey X
Y>last prey Y
Y<last prey Y
decrement x
increment x
decrement y
increment y
the cell are busy
Search for nearest free cell
Change the coordinate for the ant
The Ant is into site of catch of the last prey
Change the coordinate with the nest once
Ant not yet on the right cell

2.6 Time managing

It is necessary to manage the time matter because of the difference in time concept between human and computers. The first category sees the time as a continuous flow, while computers see the time as a discrete set of frame. The project must fill this gap in order to realise a human-friendly but efficient application.
There are several ways to fulfil this aim. I will illustrate three possibilities with pros and cons.

Threads: the single ant will be represent by single thread

- Pros:
  1. real dynamism
- Cons:
  1. difficult model to manage and foresee
  2. possibilities of conflicts on shared resource (the environment)

Asynchronous schedule:There is a central unit which schedules the ant's movements, sorting them randomly and moving one of them at time, after that all ants have made a movement the time will be incremented and the circle starts from the beginning.

- Pros:
  1. simple to implement and manage
  2. The ants' conflicts about shared resource are limited in number and complexity
- Cons:
  1. the resulting system is relatively slow
  2. the dynamism is simulated

Synchronous schedule: The schedule for each time slice lets the ants move simultaneously into the arena. They will upgrade their status and redraw the environment after every step done.

- Pros:
  1 .good parallelism
  2. the conflict through ants regarding the arena are easy to manage

- Cons:
    1. the system is slower than the threads solution
    2. the managing of collision is harder than the asynchronous solution

I chose to implement the last solution, so that I have a good compromise between simplicity and dynamism.
I called the scheduler developed for this project “Controller”.

CHAPTER 3

A Java Simulator: Project

In this chapter I will describe the structure and details about the software use for the developed application.

3.1 Classes and packages

This paragrapher contain a brief overview about the packages and related classes, for a deeper analysis you could consult the documentation attach to the source code in form of Javadoc.
There are three packages in this project:

- gui: It contain useful classes to give a graphic representation of the simulation;
- modelBlock: the core package of the project, it contains the classes used to implement the entities present in the simulation;
- test: It contains a set of tests used to verify the functionality and quality of the classes contained in “modelBlock” package and to produce the experiments results used to prove the project goals.

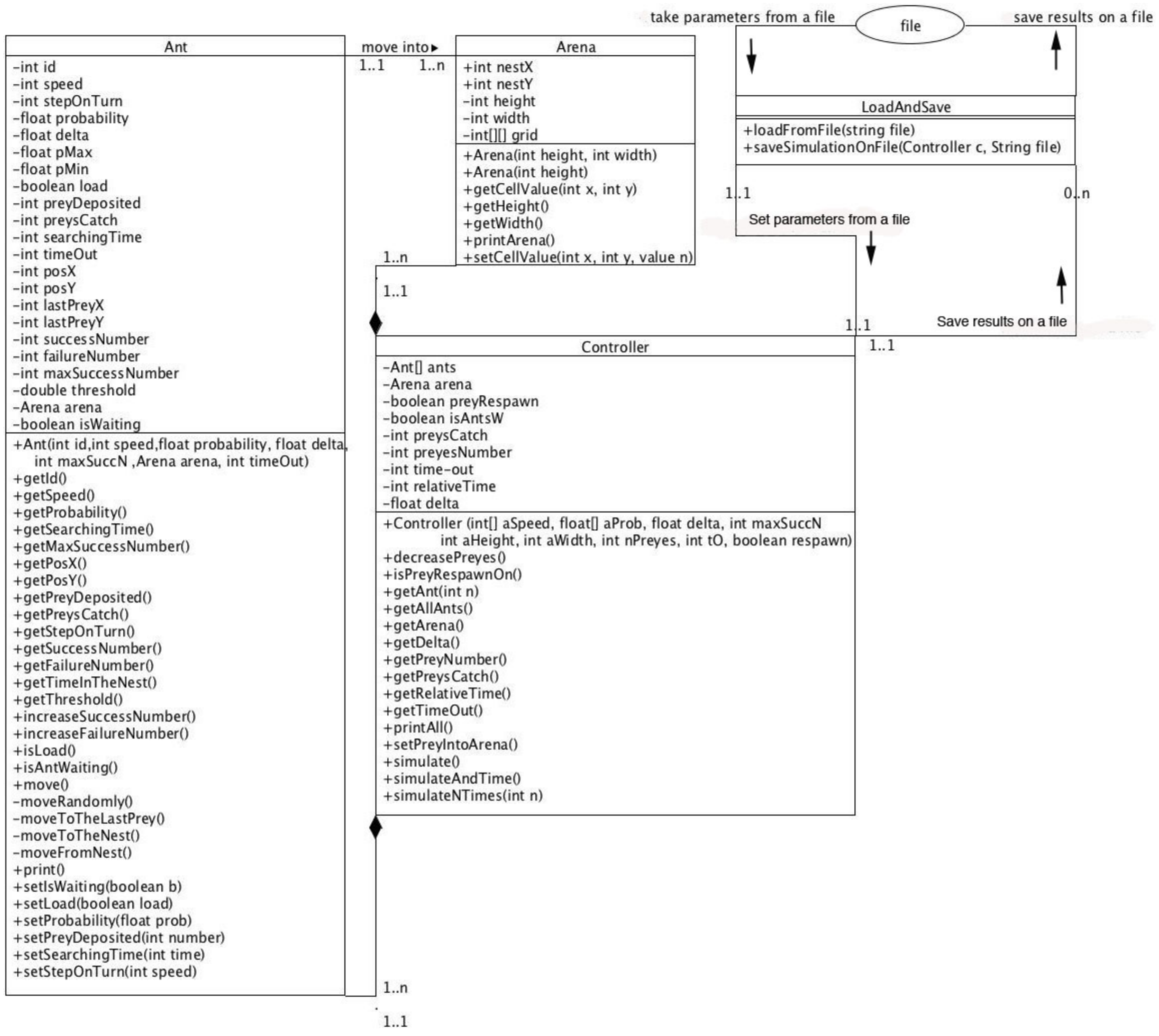
3.1.1 Gui

This package is quite primitive and underdeveloped, because I focused my attention and effort on the main features of the problem. I created one simple interface, selectable from two different looks and feel from two different files.
This interfaces was written using java swing library, the user must load the parameters list from a text file formatted in the proper way, as described successively at 3.1.2.4 in class “LoadAndSave”, contained in “modelBlock”.
Below there is an explanation of the main classes contained in this package. For more information the user could consult the relative documentation in “javadoc”.

- SimulationFromFile: Launch a simple frame (“SimulationFrame”) with a panel, “FilePanelFlow” as default or “FilePanelGrid” with a little change to the source-code, that will load a window useful to launch the simulation
- SimulationFrame: it is a simple frame with a fixed dimension
- FilePanelFlow: This Panel extends the class “JPanel”.This sub-class starts with 3 buttons and one text field:
- Input file: necessary to load a file containing the starting parameters for the simulation. This file has to be formatted as told in the documentation for class “LoadAndSave” in package “modelBlock. After the selection of a file, the starting setting for the environment is shown through a table created under the buttons.
- Run simulation once: run the simulation for a discrete time unit, the table is updated with the new situation and some information about the ant involved in the simulation are shown in a text area. This feature is useful to observe the movement of the ants in a quite user-friendly way
- Run simulation n times: button used to run the simulation for “n” discrete time unit, where “n” is the number to write in the text field next to the button. It work as the button above in term of updating and results.
- FilePanelGrid: Work as the panel above, except for the layout i.e. the order of the components.
- SimulationTable: This class represents the table used to show the experiment’s arena in a visual format. It extends the Java class “JTable”.
- SimulationTableModel: Manages the filling of the aforementioned table with the starting values.

3.1.2 modelBlock

This package contains the application's interacting entities.
The package contain the following classes:



3.1.2.1 Ant

This class represents the ant and contains all useful ants parameters and behaviours. Below I listed a brief description of ants parameters:

- speed: number of moves available for each time unit;
- probability: value used to represent the ants' probability to exit the nest and search for a prey. The starting value is set from the user and incremented or decremented as described previously in 2.5.2 ;
- delta: quantity use for probability's increment and decrement;
- pMax, pMin: respectively maximum/minimum probability's possible values. They are automatically set proportionally to the maximum number of successes, another parameter set by the user;
- isLoad: parameter use to know if the Ant is carrying a prey;
- preyDeposited: number of preys deposited by the Ant in the current turn
- preysCatch: total amount of preys caught by the Ant from the beginning of the simulation.
- timeOut: total time that the Ant could stay outside the nest before it had to come back to it;
- posX, posY: Ant's coordinates in the Arena;
- lastPreyX, lastPreyY: coordinates of the last prey caught by the Ant;
- successNumber, failureNumber: Ant's number of consecutively success/failure archives until now;
- maxSuccessNumber: maximum number of possible successes or failures that an Ant may archive;
- threshold: number to overcome to leave the nest and continue the prey retrieving task. It is automatically set proportionally to the Ant's maximum number of successes, another parameter set by the user;
- Arena: environment where the Ant has to move;
- stepOnTurn – searchingTime – isWaiting: parameters used to manage internal mechanism, respectively:
    - Ant's steps remaining for the current turn, useful parameter to know how many steps it has to do if it has to move out from its turn because it has not previously found a free space to occupy;
    - time remaining for the Ant's searching state before it has to come back to the nest;

    - a parameter used to know if the ant has to wait to continue the movement. This event occurs when the ant does not find a free space to occupy. Usually the event occurs frequently into systems with high figures for prey density and ants number. This situation is solved suspending the Ant's current turn and waiting for the other Ants' movements, then retrying with move algorithm.

Below there is a brief explanation for some relevant methods contained into this class other than the ones use to return or modify fields' values:

- move(): method used to move the Ant by a single step. Its issue is to address the Ant through the right behaviour depending by its status. To do that, this method uses the following four private methods;
    - moveRandomly(): used when the Ant is in "Search" state;
    - moveToTheLastPrey(): used after the deposit of a prey in the nest; the Ant comes back to the cell where that prey was located. If the Ant finds a prey during the path, it brings it to the nest, otherwise, when the Ant reaches its destination, it enters in "search" state;
    - moveToTheNest(): used in "Return" and "Retrieve" state i.e. when the Ant has to come back to the nest because the time-out is expired or because it has caught a prey to deposit respectively;
    - moveFromNest(): used in "Sleep" state, the Ant has to leave the nest after the "Return" state; its probability is multiplied for a random number and compared to the threshold value, if it is lower the Ant remains in the nest, otherwise it goes in "Search" state;
- print(): print all the useful information about the Ant in standard output, it is an easy way to do debugging and fixing.

3.1.2.2 Arena

This class represents the simulation's environment, the space where all the Ants will move and all the preys are set. The Arena is implemented through a matrix of integer numbers, with dimensions chosen by the user through height and width parameters. The number of resulting cells is equal to the product of Arena's dimensions, hence, the number of the cells available by Ants is:

Number of cell = Heigh* Width

The central cell is occupied by nest.
Every cell could contain one of the following values that represent a different entity:

- -1: prey;
- 0: free cell
- 0 < x < number of cell : Ant who has number “x” as id
- Number of cells: nest

This class is relatively simple, the only method where an explanation could be necessary is:

- printArena(): used to have a visual representation in standard out of the internal statement of the Arena, useful for fixing problems, giving a better understanding of the environment

3.1.2.3 Controller

This class is application’s scheduler, its functionality are:

- Create the simulation environment :
    - Create Arena and filling it with preys;
    - Create Ants with the parameters set by user.
- Manage:
    - the Ants’ movements;
    - the prey re spawn working;
    - the time passing;
    - Harvest simulation’s datas.
      Several of Controller’s fields in fact are useful to gain information about the simulation.

Below there is a brief explanation for some relevant methods contained into this class other than the ones use to return or modify fields’ values:

- printAll(): This method reports all the relevant values contained in the system’s entities.

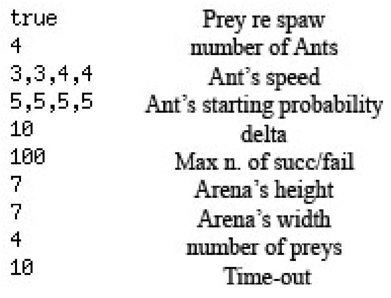
- SetPreyIntoArena(): Use to fill the Arena with preys at the simulation 's start and every time an Ant deposit a prey into the nest if the prey re spawn option is active. This method randomly choose an Arena's cell and check her content, if this one is free, the method set a prey, otherwise search for another cell;
- simulate(): runs the simulation for one time unit. The Controller lets all Ants to move than increment the discrete time by one unit;
- simulateAndTime(): work as the aforementioned method, but returns the amount of real time passed during that simulation;
- simulateNTimes(int n): this method runs the simulation for a number of time unit set by the user with parameter "n" and return the real time duration for the simulation.

3.1.2.4 LoadAndSave

This class is a library which contain two static methods use to interact with I/O (input/output) through the file system.
It is useful to load the requires system's parameters and save the simulation's results into a consistent form

- loadFromFile(String file): this method uses the list of parameters contained in the file with the name set by "file" parameter and return a Controller instance set with user's parameters. The file have to be formatted in the following way to be read successfully by this method:
    - Prey re spawn: set "true": to activate this option, "false" otherwise;
    - Ants' number;
    - Ants' speed: As a list of integer numbers separated by a ",";
    - Ants' starting probability: As a list of integer numbers that represent the decimal part of the number, all this value have to be separated by ",";
    - Delta;
    - Max number of success (and failures) ;
    - Arena's height;
    - Arena's width;
    - Number of preys;
    - Time-Out.

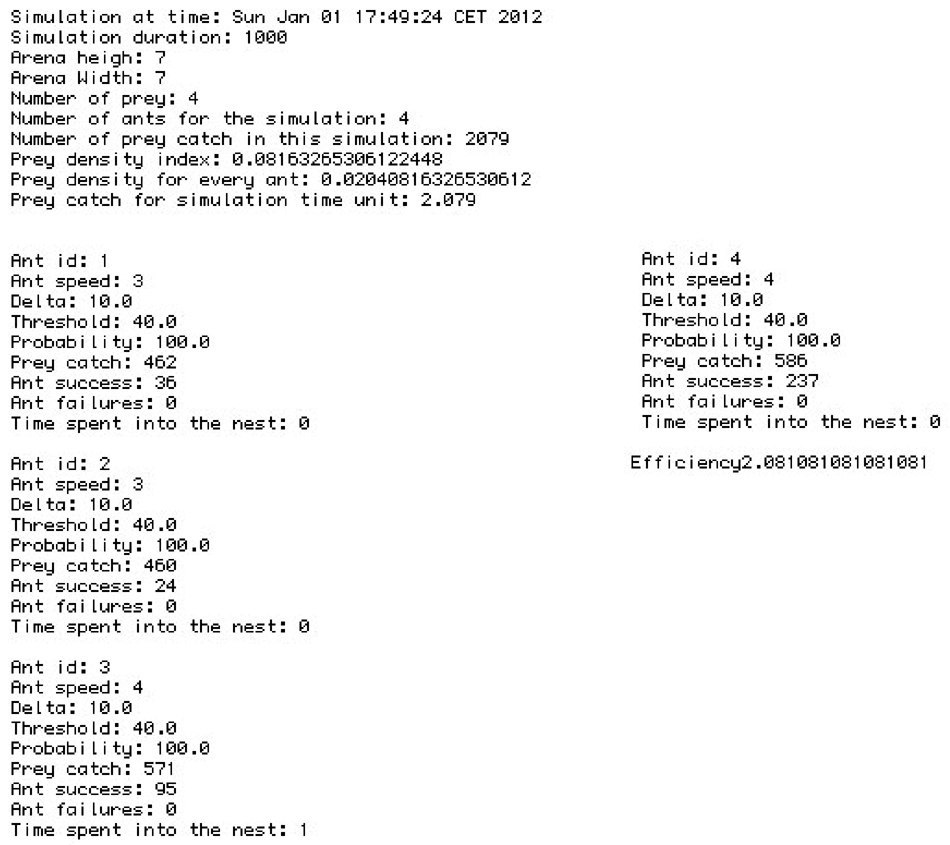
Example of well formatted text file:



```
true        Prey re spaw
4           number of Ants
3,3,4,4     Ant's speed
5,5,5,5     Ant's starting probability
10          delta
100         Max n. of succ/fail
7           Arena's height
7           Arena's width
4           number of preys
10          Time-out
```

- saveSimulationOnFile(Controller c, String file): Useful method to save the simulation's results obtained using the Controller "c" on the file with name or path set by "file".".

Here an example of the display for a simulation with 4 ants:



```
Simulation at time: Sun Jan 01 17:49:24 CET 2012
Simulation duration: 1000
Arena heigh: 7
Arena Width: 7
Number of prey: 4
Number of ants for the simulation: 4
Number of prey catch in this simulation: 2079
Prey density index: 0.08163265306122448
Prey density for every ant: 0.02040816326530612
Prey catch for simulation time unit: 2.079


Ant id: 1
Ant speed: 3
Delta: 10.0
Threshold: 40.0
Probability: 100.0
Prey catch: 462
Ant success: 36
Ant failures: 0
Time spent into the nest: 0

Ant id: 2
Ant speed: 3
Delta: 10.0
Threshold: 40.0
Probability: 100.0
Prey catch: 460
Ant success: 24
Ant failures: 0
Time spent into the nest: 0

Ant id: 3
Ant speed: 4
Delta: 10.0
Threshold: 40.0
Probability: 100.0
Prey catch: 571
Ant success: 95
Ant failures: 0
Time spent into the nest: 1

Ant id: 4
Ant speed: 4
Delta: 10.0
Threshold: 40.0
Probability: 100.0
Prey catch: 586
Ant success: 237
Ant failures: 0
Time spent into the nest: 0

Efficiency2.081081081081081
```

3.1.3 Test

This package contain 4 kind of classes:

- AntTest<x>: A kind of test I used to verify the good working of Ant class
- ArenaTest: used to test the simple Arena class
- ControllerTest<x>: That test was used to verify the good working of Controller class
- Experiment<String>: Classes used to test the entire system and obtain data from the application

3.2 Behaviour

I will resume the entities' behaviours using a pseudo-code representation.

3.2.1 Ant

The main Ant's activity is to move into the Arena, so I will illustrate every step in prey retrieval

move()

```
If the simulation will starts now ->  relative time = 0
    move randomly into the Arena searching for a prey  -> moveRandomly()
else
    if Ant is in the nest
        if it has come back to the nest last time slice without a prey
            increment failures number -> failureNumber ++
            set successes number to 0 -> successNumber=0
            set probability as max[pMin, probability-delta*(failure number)]
        fi
        moveFromTheNest()
    else
        if the Ant has a prey to deposit or its time-out was expired
            come back to the nest -> moveToTheNest()
        else
        if the Ant has recently deposited a prey and has to come back to the site of
        capture
            go to the site where the last prey was -> moveToTheLastPrey()
        else
            move randomly into the Arena searching a prey -> moveRandomly()
            fi
        fi
    fi
fi
```

moveRandomly()

```
search for a prey in adjacent cells
if the Ant finds a prey
    Catch the prey -> setLoad(true)
    Remove the prey from the Arena -> setCellValue(x prey,y prey,0)
    Come back to the nest to deposit the prey -> moveToTheNest()
else
    while the Ant has not done her step
          select randomly one of the eight adjacent cells
          if the selected cell is free
            move into selected cell and changing coordinate -> posX=new x;
            posY=new y
            refer the change to the Arena -> arena.setCellValue(x,y,id);
            if this was the last Ant's step for current turn and she has not caught a
            prey
                decrease searching time -> setSearchingTime(searchingTime-1)
               fi
               step done!
            else
                increment number of times the ant tried to move into a free cell ->
                n++
                If it could not find a free cell for too many times -> n==10
                    wait for the movement of the others Ants than move again
            fi
        fi
    fi
fi
```

moveToTheLastPrey

```
check for a prey in adjacent cells
if the Ant finds a prey
    Catch the prey -> setLoad(true)
    Remove the prey from the Arena -> setCellValue(x prey,y prey,0)
    Come back to the nest to deposit the prey -> moveToTheNest()
else
    compare the actual position to the destination and choose the shortest path
    if the selected cell is free
        if the reached cell is the destination
            change the coordinate of the last prey caught with the nest ones
        fi
        move into selected cell changing coordinates -> posX=new x; posY=new y
        refer the change to the Arena -> arena.setCellValue(x,y,id)
    else
        find a suitable free cell even if it is not the optimal one, so compare the actual
        position to the destination and choose another good path
        move into selected cell changing coordinates -> posX=new x; posY=new y
        refer the change to the Arena -> moveToTheNest()
    fi
fi
```

moveToTheNest()

```
compare the actual position to the destination and choose the shortest path
    if the selected cell is free
        if the reached cell is the nest
            if the Ant has a prey
                deposit the prey -> setLoad(false)
                increment the number of preys catch -> preyCatch ++
                increment the number of successes -> successNumber ++
                set failures number to zero -> failureNumber=0
                change the probability according to the algorithm
                set probability as min[pMax, probability+delta*(success number)]
            fi
        if
        move into selected cell changing coordinates-> posX=new x; posY=new y
        refer the change to the Arena -> arena.setCellValue(x,y,id)
    else
        find a suitable free cell even if it is not the optimal ones, so compare the
        actual position to the destination and choose another good path
        move into selected cell changing coordinates -> posX=new x; posY=new y
        refer the change to the Arena -> arena.setCellValue(x,y,id)
```

moveFromNest()

```
if the Ant has come back with a prey -> isLoad( ) == true
    go back to the cell where this prey was -> moveToTheLastPrey()
else
    generate a random number 0< X <10
    if (probability * X > threshold)
        move randomly out from the nest searching a prey  -> moveRandomly()
    else
        wait in the nest incrementing the amount of time passed in the nest
    fi
fi
```

3.2.2 Controller

After the environment’s creation, the main Controller’s issue is to run the simulation, letting the ants movements

simulate()

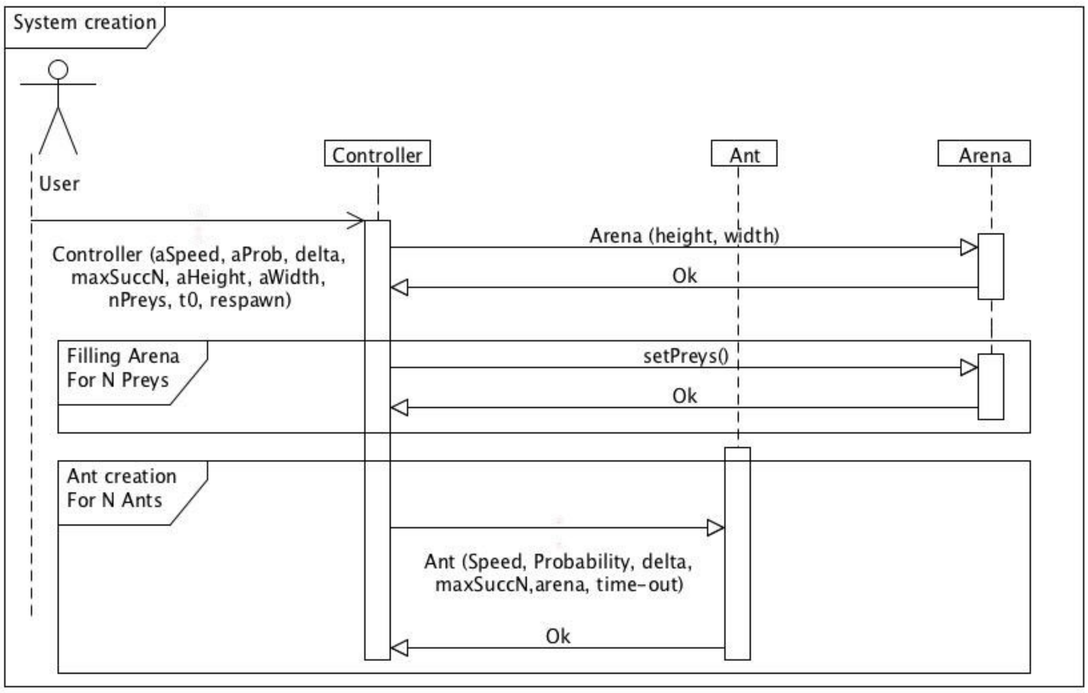
```
for every ant in the experiment -> ants.lenghts
    for the full speed of the ant -> ants[i].speed
        the ant does a step -> move()
    if the ant is waiting for the other ants’ movement because she has not found a free
    cell
        tell this to Controller
    if the Ant has caught one or more preys -> preyDeposited > 0
        if the replacement of the preys is set on -> isPreyRespawnOn()==true
            replace catch preys
        fi
    fi
fi
if there are ants who are waiting let’s move theme
    for every ant which is waiting
        for the remained ant’s speed
            the ant does a step -> move( )
        if the Ant has caught one or more preys -> preyDeposited > 0
        if the replacement of the preys is set on -> isPreyRespawnOn()==true
            Replace catch preys
        fi
    fi
fi
increment discrete time -> relativeTime ++
```

3.3 Interaction

In this paragrapher i willl show more clearly in a schematic way the interactions between system's entities

3.3.1 Simulation setting

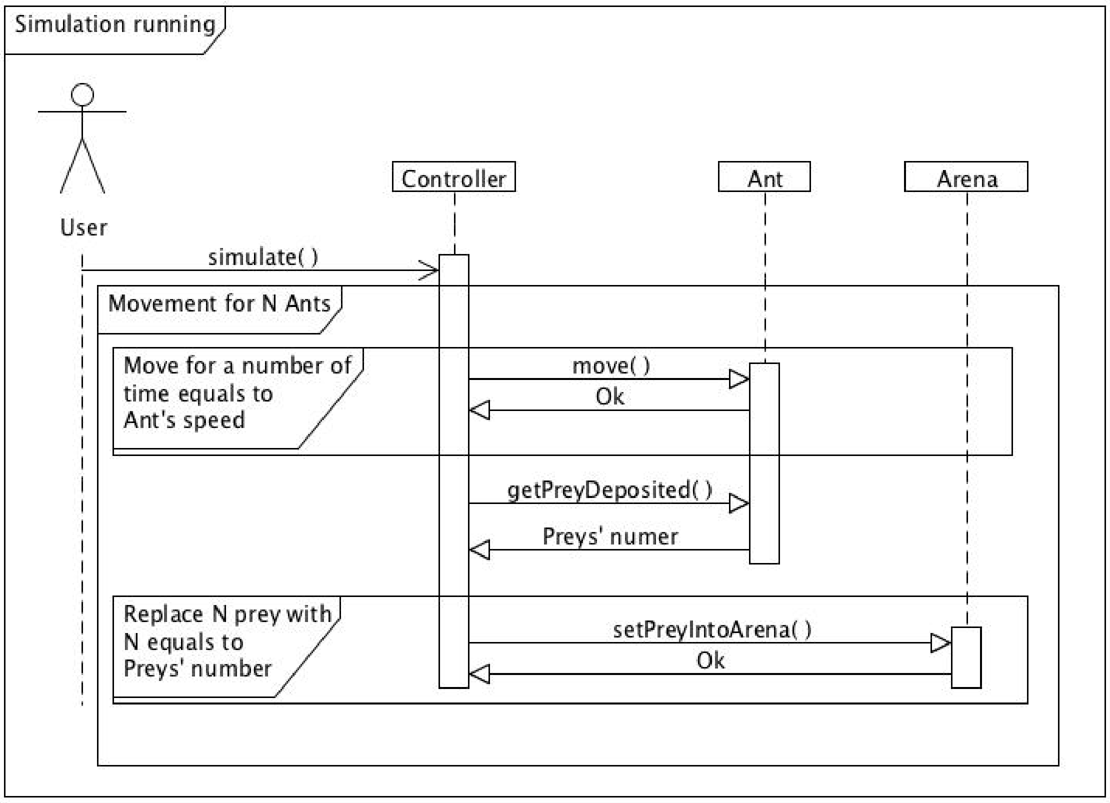
These are the interactions between the system's elements in order to start the simulation


System creation
User
Controller
Ant
Arena
Controller (aSpeed, aProb, delta, maxSuccN, aHeight, aWidth, nPreys, t0, respawn)
Arena (height, width)
Ok
Filling Arena For N Preys
setPreys()
Ok
Ant creation For N Ants
Ant (Speed, Probability, delta, maxSuccN,arena, time-out)
Ok

The Controller takes the parameters, set from users through I/O, and generates the other objects.
He builds the Arena using the proper height and width parameters, then fills it with the right number of preys using the method "setPreys()" one time for every prey.
After the Arena's creation the Controller creates the Ants, creates an internal array of the proper length in order control Ants, then creates them using the user's parameters

3.3.2 Simulation

I will show the entities’ interactions during the use of “simulate( )” function, used to run the application.



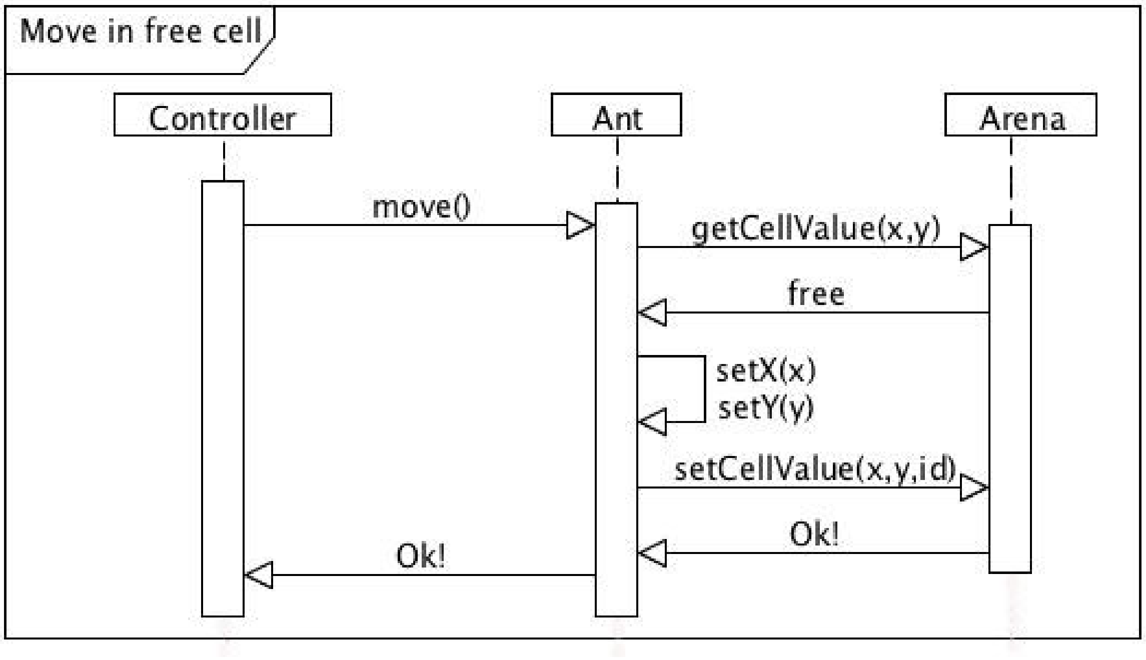
For each “simulate” ‘s instance, the Controller lets to the ants the possibility to move into the Arena for a number of times equal to their speed, then the Controller will check if the selected Ant has caught some prey during the current time unit. If it did and the “prey re spawn” option is active, the captured preys will be replaced.

3.3.3 Ants movements

The ants move independently into the Arena, but, in order to easily manage the concurrency, the Controller acts as scheduler and gives to the ants the possibility to do their turn in certain moments.
I have previously illustrated the "move()" function working; now I will show the interactions between the entities during the function execution.

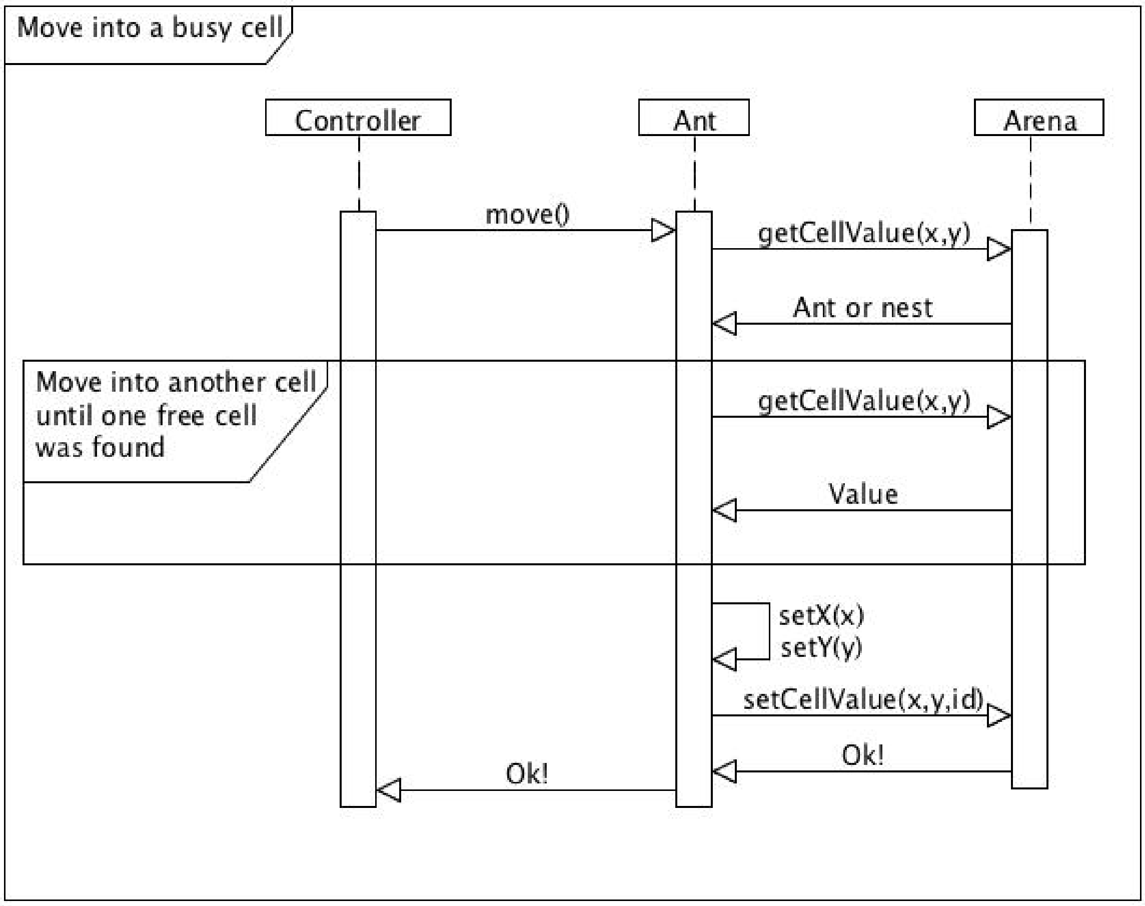
3.3.3.1 Move into a free cell

Below is represented the behaviour of an Ant which moves into a free cell and the ant's interactions with the Controller and the Arena.



3.3.3.2 Movement into a busy cell

Below is represented the behaviour of an Ant which encounters a busy cell while it was searching for a free one and the ant's interactions with the Controller and the Arena.



CHAPTER 4

Experiments

In this chapter i will illustrate the experiments done to prove the validity of work done

This chapter is dedicated to the demonstration of the starting hypothesis through the report of experiment results. The experiments, like the source code writing, were made with Eclipse, a software environment used to run the binary file written in java.
During the thesis' development, I ran a huge number of simulations, in order to verify step by step entities' features and fix all bugs or errors in short time. Through this procedure, composed by project, writing and testing, I successfully developed the application using a correct design and project methods as declared by the classical software engineering patterns.
The number of experiments I did on that application are enough both in number and accuracy ,but I will propose only a significant part of them. This set is designed to show some of the developed project's key property.
I divided the experiments in three main blocks. For each of them I maintained constant all variables except one, in order to see the variable's effect on the system. These variables are:

- Prey density
- Ant's speed
- Time in the nest

4.1 Prey density experiments

Issue: prove model's flexibility showing the correlation between the number of preys caught by Ants and the number of preys in the environment

I considered prey density as the mix of two parameter:

- Preys number

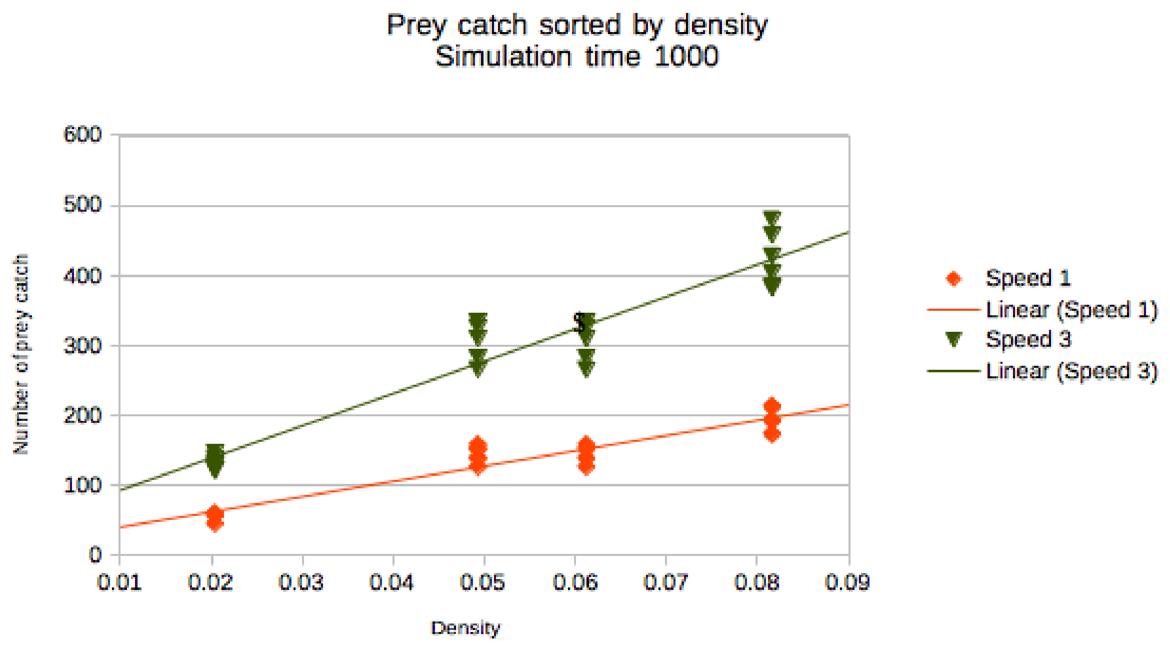
- Arena’s dimensions

composed the “prey density” parameter as follows:

Preys number/Cells number = Preys number/Arena’s high*width

For these experiments I used a little set of data obtained through some simulations involving four Ants, two of them with a speed equal to 1, while the other two with a speed equal to 3. Every simulation was done for 1000 ticks. During different simulations I used different prey density values. Below are reported the results.

On the x-axis are reported the different prey density values, while on the y-axis are reported the different preys catch number. In red are marked the number of preys caught by the ants with a speed equal to 1, in dark green the same quantity for ants with a speed equal to 3. I used four different values for prey density, in order to archive different significant results.



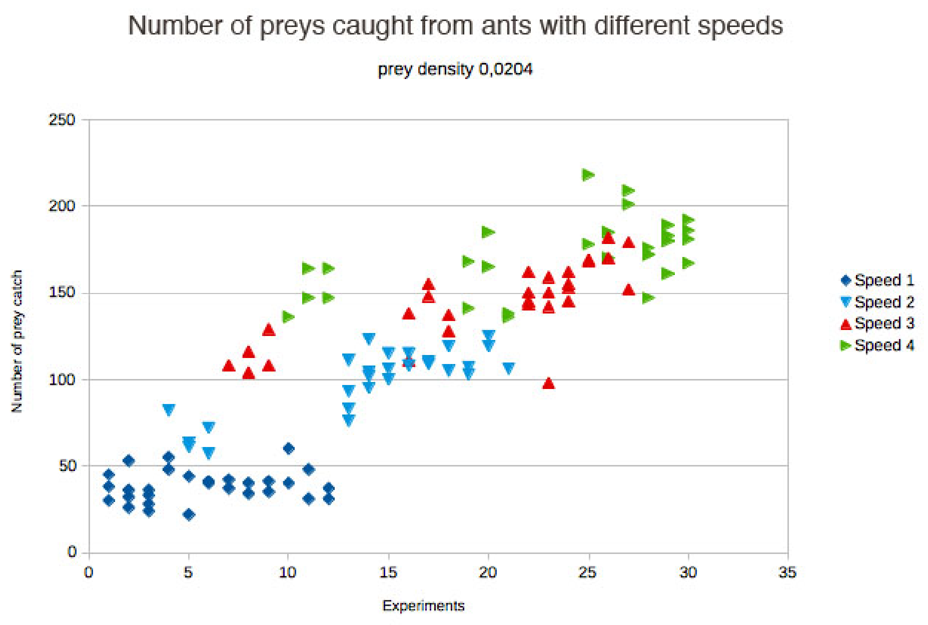
Results
The correlation between number of prey caught by ants and prey density is clear from the chart.(le due parti della frase vanno invertite)
Other results, better explained in the following chapter, is the correlation between Ants' speed and number of preys caught.

4.2 Speed experiments

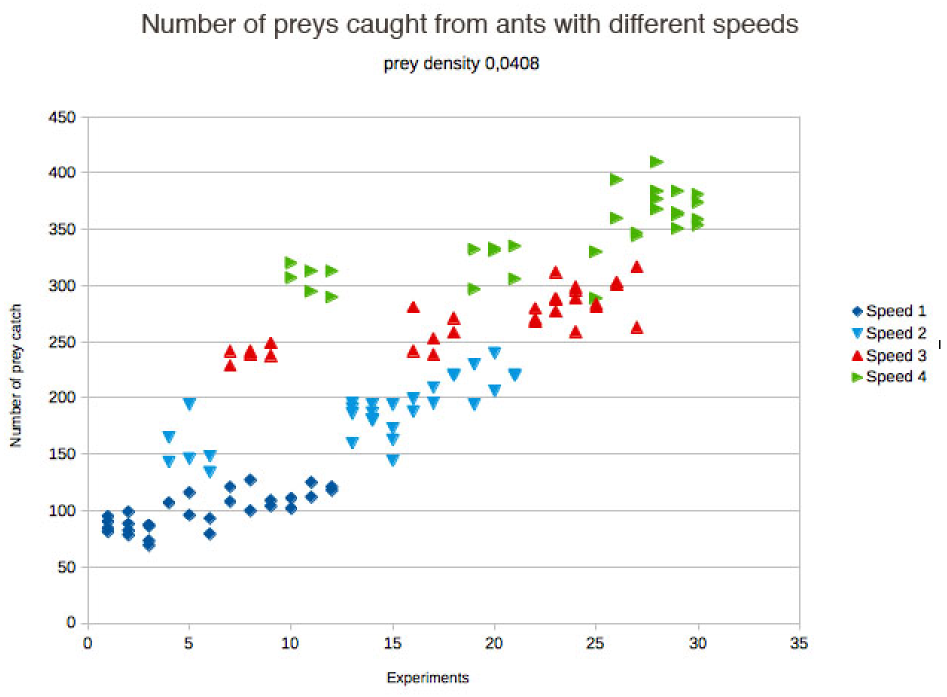
Issue: prove correlation between ants' performances and Ants' speed, as faster Ants catch more preys.

Speed is one of Ant's main features. In facts, the Ant's performance and whole system's one, is mainly influenced by this parameter.
I designed these experiments using 3 different prey density, in order to observe similar behaviour independently from this value. For every density I used the same data. Every experiment has the duration of 1000 tics. For each experiment I used a different set of four Ants, all with the same speed or with the same speed in pairs (for example either four Ants with a speed equal to 1 or two Ants with a speed equal to 1 and two with a speed equal to 2).
In the chart shown below, I summarised that data dividing them between the different Ants' speed. For every value on the x-axis there are four values on the y-axis; each of them is the number of preys caught by an Ant with a certain speed during the experiment.
For every Ant's set I did 3 experiments. I started with four Ants with speed 1, than with two Ants with speed 1 and two with speed 2 and so on.

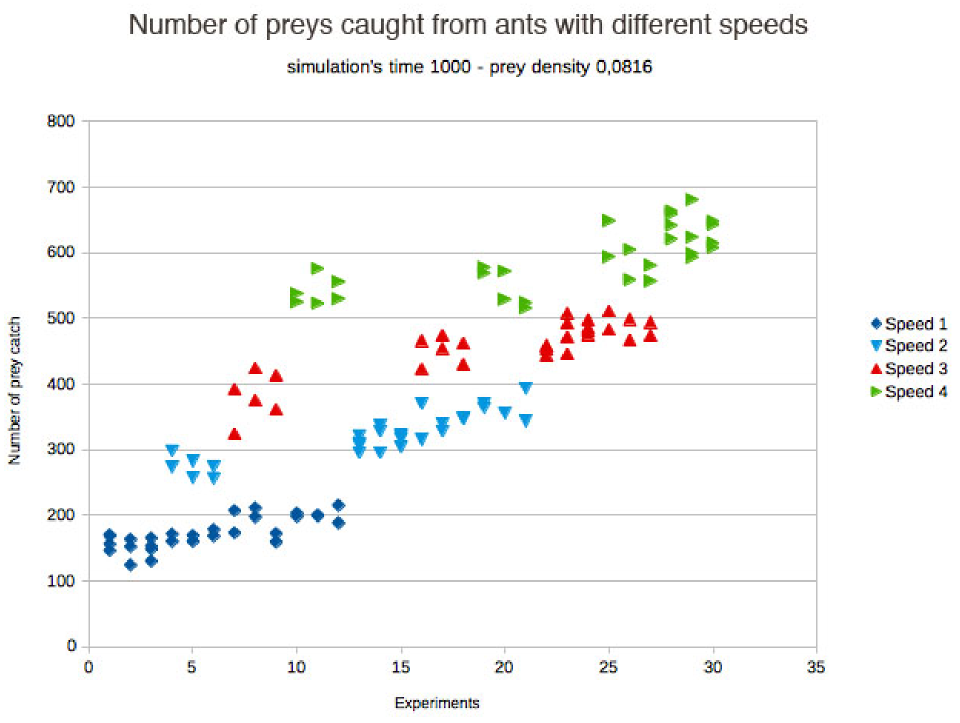
Experiments with prey density equal to 0,0204



Experiments with prey density equal to 0,0408



Experiments with prey density equal to 0,0816

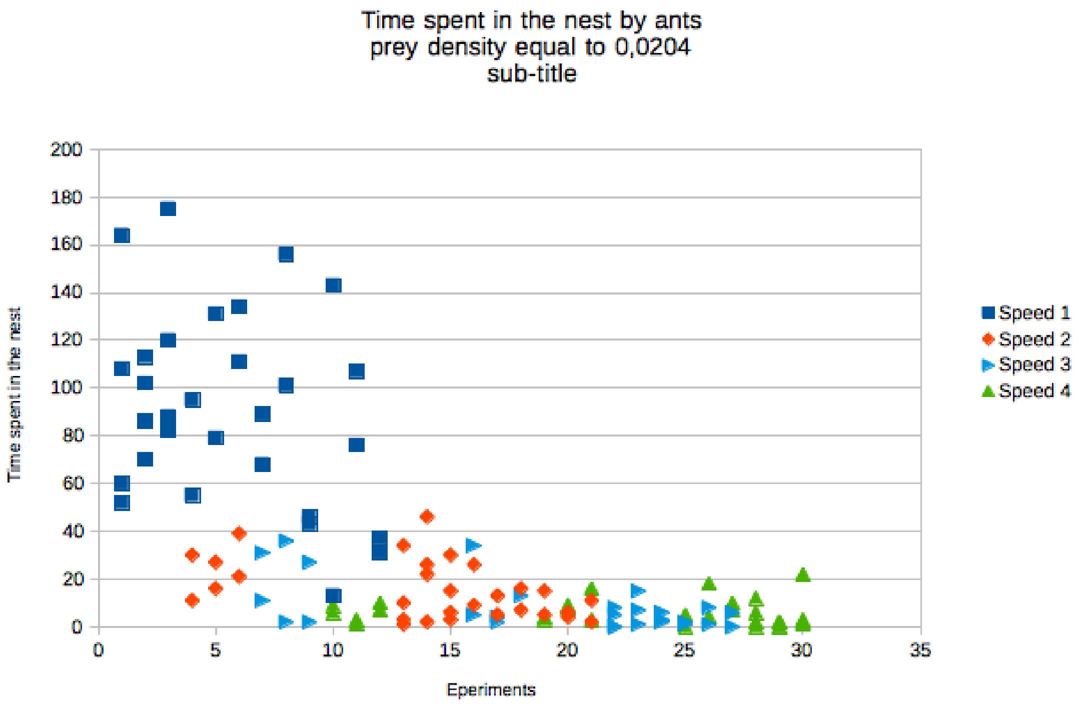


<u>Results:</u>
From the data and charts, it's clear that faster Ants catch more preys than slower ones. However this is not the only conclusion, in facts, it is also clear that the Ant's performance is also influenced by the mates' speeds. If an Ant with a given speed works with faster Ants, it will catch more preys than in a simulation with slower mates or mates of the same speed. This is clear watching the charts that have a greater gradient into the end of the axis, where there are teams with greater speed.

4.3 Time in the nest experiments

Issue: prove model's flexibility and ants' specialisation created by division of labour.

The experiments' design is set like the previous one, the difference being the quantity represented by the y-axis, which in this case is the time spent in the nest by ants.
Like the previous experiments, I maintained all the variables constant except for the Ants' speed.
I used four ants for every experiments, this agent's speed is always equal between all of them or in pairs.



Results:
The experiments prove the starting hypothesis: slower ants spend more time into the nest, doing other works than prey retrieving, while faster ants usually stay outside the nest nearly the whole time, searching for prey.

Conclusions

The goals for this work was to illustrate the concept of Swarm Intelligence, the characteristics of the system which implements this idea and how it could be interesting for two different scientific fields like computer-science and biology.
To write this work i was inspired by Labella(2006)'s work, who has simulated the ants' behaviour both with virtual agents and real robots, reporting his experiments results to prove his starting hypothesis.
Differently from Labella i used only virtual agents. I proved the starting hypothesis through several experiments using different sets of ants with different characteristics. Changing the ants' parameters from one experiment to another prove the validity of the developed application even in different situation, not only for a specific case.
Thanks to three series of experiments i prove the starting hypothesis:

- The number of preys caught during the experiments is proportional to the number of prey avaiable in the environment;
- Faster ants catch more preys
- The ants who catch more preys became specialist for preys retrieving task, hence they spend most of their time outside the nest searching for food source. Vice-versa, the ants who catch less preys usually spend more time in the nest, dealing with other works

I had obtained a good model, it is flexible, realistic and suitable for other experiments. I noticed one drawback to modify in future reseal or version. The threshold mechanism i had implemented does not lead to a strength specialisation mechanism especially for higher density, where even ants with lower speed use to stay outside the nest more times comparing with lower density experiments
There are several possible changes to upgrade the application, for example:

- Change the threshold value dynamically
- Change the nest dimensions and shape
- Simulate natural obstacles and difficult terrains for different cells, in that way an ant could cover a cell with difficult terrain only with more step
- Use ants with different “sight” that can see obstacles and prey in their surrounding and not only in adjacent cells
- The possibility for the prey to escape from ants.

Even if the project could be optimized, the final result reach the prefixed goals and give a good representation of a flexible model base on Swam Intelligence technic.

References

- Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). Swarm intelligence—from natural to artificial systems. Oxford: Oxford University Press.
- Bonabeau,E. , Theraulaz, G. , Deneubourg J.-L. , Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies, Proc. Roy. Soc. London B 263 (1996) 1565–1569.
- Camazine, S., Deneubourg, J. L., Franks, N. R., Sneyd, J., Theraulaz, G., & Bonabeau, E. (2001). Self-organization in biological systems. Princeton: Princeton University Press
- Deneubourg J.-L., Aron S.,Goss S., J.-M. Pasteels, The self-organizing exploratory pattern of the Argentine ant, J. Insect Behav. 3 (1990)
- Detrain, C. and Deneubourg, J.-L.1997. Scavenging by Pheidole Pallidula: A key for understand- ing decision-making systems in ants. Animal Behaviour 53, 537–547.
- Dorigo Marco, Bonabeau EricTheraulaz , Guy Ant algorithms and stigmergyc a IRIDIA, Université Libre de Bruxelles,Université Paul Sabatier, Toulouse 2000.
- Grassé P.P., La reconstruction du nid et les coordina- tions interindividuelles chez bellicositermes natalensis et cubitermes sp. La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs, Insectes Sociaux 6 (1959) 41–81.
- Garnier, The biological principles of swarm intelligence-2007 © Springer Science Business Media, LLC 2007
- Holldobler, B. and Wilson, E.1990.The Ants. Springer Verlag, Heidelberg, Germany.
- Krieger and Billeter [The call of duty: self-organized task allocation in a population of up to twelve mobile robots, Robot. Autonomous Systems-30 (2000) 65–84.]
- Labella H. Thomas and Dorigo Marco Division of Labor in a Group of Robots Inspired by Ants' Foraging Behaviour 2006 IRIDIA, Universite Libre de Bruxelles, Brussels, Belgium
- Theraulaz, G., & Bonabeau, E. (1995a). Coordination in distributed building. Science, 269(5224), 686–688.
- Theraulaz, G., & Bonabeau, E. (1995b). Modeling the collective building of complex architectures in social insects with lattice swarms. Journal of Theoretical Biology, 177, 381–400.
- Thiemo Krink Swarm intelligence introduction