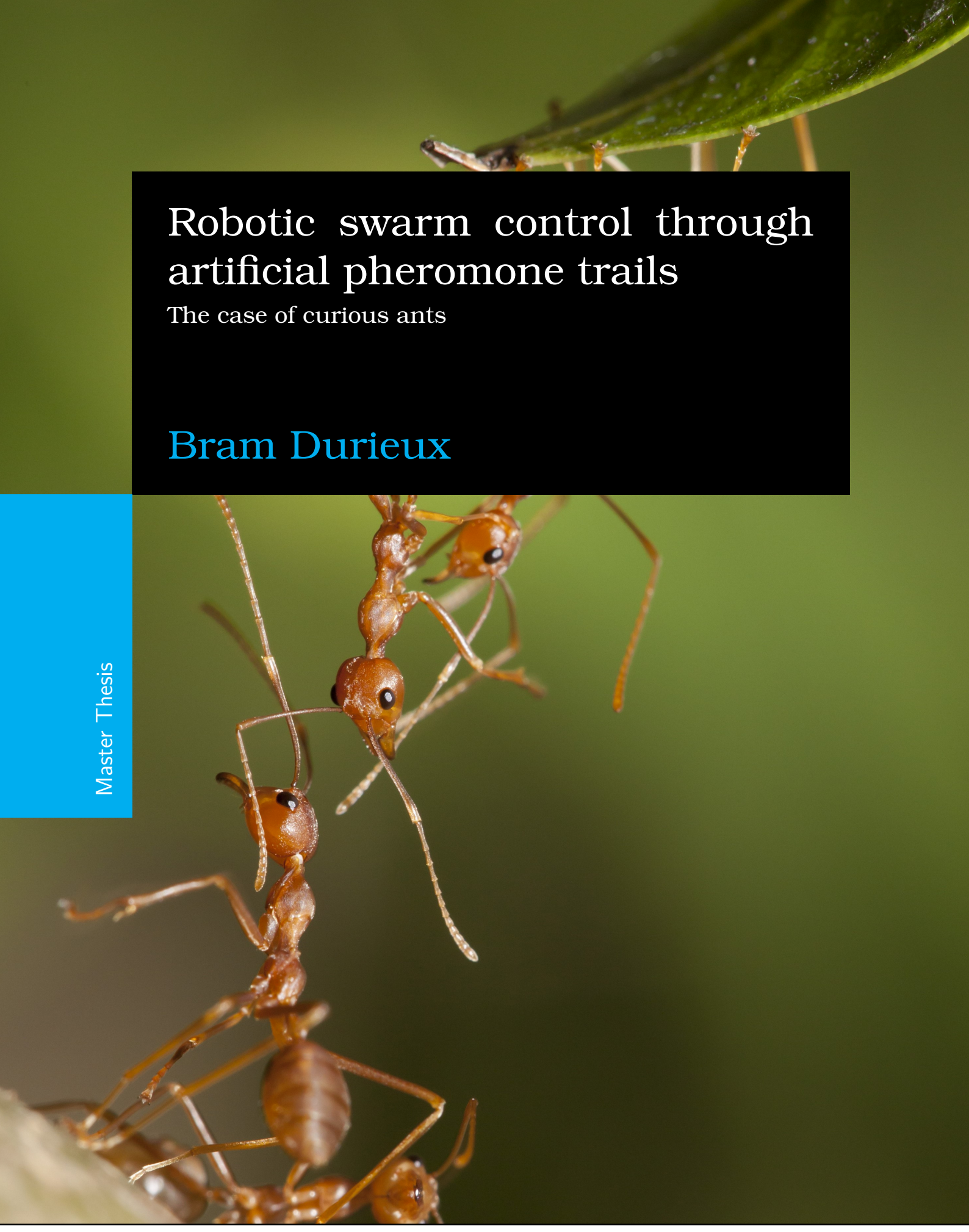


Robotic swarm control through artificial pheromone trails

The case of curious ants

Bram Durieux

Master Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Robotic swarm control through artificial pheromone trails

The case of curious ants

For the degree of Master of Science in Systems and Control at Delft University of Technology

Bram Durieux

June 13, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Cover image [1].

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Robotic swarm control through artificial pheromone trails

by

Bram Durieux

in partial fulfillment of the requirements for the degree of

Master of Science.

Dated: June 13, 2019

Supervisor:

Manuel Mazo Jr.

Readers:

Daniel Jarne Ornia

Javier Alonso Mora

Kim Batselier

Abstract

The extraordinary capability of swarming ant species in route finding and foraging efficiency through trail trail development has been studied for many years. Scientists have been able to capture the behavior of individual ants in control algorithms and used the resulting artificial swarm of ants to solve difficult problems. These problems include function optimization, sorting, clustering and route finding. So far, the route finding capabilities, where agents solely rely on local available information, have only been demonstrated in virtual discrete domains.

The aim of this thesis is the development of an agent based control method that allows a swarm of robots to establish a trail between a source and target, placed in an unknown domain. The method solely relies on information locally available to the agent. No direct communication among the agents is necessary as agents mark the environment to relay information. Simulations are performed in a Python and C-based computer program, developed in tandem with this study. A versatile controller is presented that, in conjunction with suggestions for the swarm size and pheromone characteristics, can be implemented on real robots to explore unknown environments in a multitude of scenarios.

Acknowledgments

I wish to express my gratitude to everyone frequently present in our weekly meetings, with Professor Manuel Mazo Jr. in particular. Thank you for your patience, feedback and guidance throughout this journey. I truly believe our discussions and consequent feedback elevated this project to an other level.

To my partners in crime at the DCSC laboratory. We have kept each other focused, shared our skills and drank a lot of coffee. Miguel and Arjan, thank you for the help with programming the Elisa3 robots.

Anneloes, you've been wonderful and continue to amaze me. The illustrations you have provided, feedback and mental support means the world to me.

Summary

The concept of stigmergy allows many swarming insect species to handle complicated tasks. Among swarming ants, this concept leads to foraging trails between their nest and food sources in the surroundings. The capability of these ants has been studied for many years. In studies, researchers have modeled the ant response to a marking agent called pheromone, in an attempt to explain the formation of efficient foraging trails. Many applications have since been derived.

One such application is to let a swarm of ant-based, virtual agents, to find the shortest route in discrete domains. The control objective is to let the agents solely respond to the presence of pheromone. Consequently, this concept was applied to a swarm of robot for the task of route finding. Although these methods work well, up until now, problems regarding the efficiency of the swarm persist throughout continuous domain experiments.

In order to achieve a decentralized, stigmergy based control method for robotic ants, a software simulation platform was designed. With the software, two probabilistic control methods are analyzed. The first one is simple where the response of the agent is a direct consequence of the difference in pheromone present at the left and right of the agent and a source of noise. This method has its flaws, mostly as the noise prevents agents from following a pheromone trail. However, when the noise influence is decreased, agents are unable to find the nest in the first place.

A second control method combats the problems by having the agent response to the difference in pheromone relative to the total pheromone quantity in its vicinity. This way, both efficient exploration of the domain and exploitation of the food source is possible. It is demonstrated, that when agents have knowledge about their position relative to the nest, the results are even better.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ever since micro processors are available to the public, computing power has gotten cheaper each year. Nowadays small single board computers, the size of a large insect, harbor more computing power than scientific mainframe computers did some decades ago. These small, so called systems-on-a-chip, can be packaged on a mobile framework with sensors and actuators yielding a tiny and inexpensive robot. It is interesting to see what one can achieve when putting a swarm of these robots together.

The challenge of handling such a swarm of robots is controlling it. Coordinating a whole lot of swarming robots simultaneously, generally poses a demanding challenge on the communication infrastructure or possibly a central controller. One can write sophisticated decentralized controllers to mitigate the computing power problem at the expense of an extra burden on the communication infrastructure. An other way to control such a swarm, is by acknowledging that in nature, swarming insect species rely on their swarm intelligence without advanced communication systems. By mimicking how swarming ants use indirect communications to relay messages throughout the swarm, one may obtain a low cost yet highly capable and robust system for specific tasks. As an ant's limited brain capacity acts as a form of simple decentralized controller.

One such task is the exploration of and navigation through unknown environments. Imagine one such environment where there is no global satellite navigation system available. Examples areas are: indoors, shaded by tall obstacles (mountains, skyscrapers), uncharted terrain (deep in the jungle) or extraterrestrial (mars, the moon). For ants, this is their reality. Successfully navigating such an environment, could be possible by applying the concept of stigmergy [2] to the artificial robots.

Leveraging nature's ways of solving a problem in this field is not new. For decades scientists and biologists have studied the behavioral patterns of various species of termites and swarming ants. These studies have revealed individual agent-based rules that yield collective swarm behavior for navigating mazes and exploiting food sources. Scientists have used these findings to develop artificial numerical optimization strategies based on the concept of stigmergy [3] and engineers have applied the characteristic method of

marking the environment on robotic swarms that carry out complicated tasks without explicit communications. In 2017, a former TU Delft student, Daniel Jarne Ornia under supervision of Dr M. Mazo Jr. and Dr X. Hu formalized an agent based control law for navigating a discretized 2-dimensional environment with guaranteed convergence[1] to an optimal solution [4].

In this master thesis a simulation framework is presented which is used to develop an ant-inspired control law for simulated ants (agents) in a continuous domain. Of particular interest is to what extend the findings of D. Jarne Ornia MSc thesis correlate with a continuous domain. A proposed extension is presented to demonstrate the resulting algorithms on real robots. To achieve this, a brief description of the related background material and an overview of the goals for this thesis is presented in Chapter 2. Chapter 3 contains a formal description of the bundled concepts and a mathematical framework for this thesis and an elaborate description of the simulation, its agents and domain. With the use of a computer simulation program, two control methods are tested for multiple scenarios in Chapter 4, followed by a discussion on the results in Chapter 5. Ultimately, a reflection of all targets and achievements is presented in Chapter 6 and possible new research directions are discussed in Section 7-5. The software package is described in Appendices A and B.

[1] under certain assumptions

Chapter 2

Background, goals and expectation

With the aim of creating a working simulation of a robot swarm, the related theory is summarized in this chapter. Based on the of the literature study [5], the scope of this thesis is outlined at the end of this chapter. Among the interesting findings are the problems others have encountered with pheromone based navigation. Next, a preceding study from a fellow student who accomplished stigmergy based exploration and exploitation in a discrete domain is briefly analyzed. Finally, the findings of others who have applied the concept of stigmergy to real robot navigation.

2-1 Stigmergy

Some animal species have a special way of communication, by marking the environment with a marking agent in such a way that others can respond to it. By doing so, no direct interaction between animals is required, thus the intelligence for the processing of verbal communication is not required. Even with the lack of direct communication, by merely responding to the presence of a marking agent, complex collaborative behavior is exhibited by many different insect species. Nowadays, we refer to this process as 'stigmergy'.

The concept of stigmergy was introduced by mr Pierre-Paul Grassé [2]. It covers the collective behavior of predominantly swarming organisms through indirect ways of communication. In many species of swarming ants such as the Argentine ant *(Linepithema humile)* (commonly known as the Argentine ant), stigmergy allows the colony to swiftly explore environments and exploit the food sources efficiently.

Swarming ants use the concept of stigmergy to a great extend. By collectively marking the environment with pheromones (a volatile chemical compound, the marking agent), complex networks of trails, linking the nest and discovered food locations emerge. Ants simply respond to the presence of pheromone and are inclined to walk in the direction

of the highest concentration of pheromone while depositing even more, leading to trail reinforcement. Consequentially, this behavior leads to so called foraging networks [6].

The concept of pheromone based stigmergy, with its characteristics of indirect communication among agents, reinforcement (positive feedback) of 'good' trails and evaporation of the 'bad' ones, can be used to solve a number of artificial problems.

2-2 ACO research directions

Probably the best known example of how the concept of stigmergy is used to solve artificial problems is Ant Colony Optimization (ACO) [3, 7]. Here, a population of agents (ants) solve an optimization problem, posed as a graph, on an iterative basis. ACO is a broad category of swarming ant inspired meta-heuristics. Many different variants of ACO have successfully been applied to NP-hard problems. Most famously, the Traveling Salesman Problem (TSP), a simple graph where vertices are connected with links.

Eventually, all ACO algorithms are built on the key components of stigmergy: agents reinforce good solutions with pheromone due to a preference bias, while less optimal solutions are gradually being forgotten due to evaporation of the pheromone.

2-2-1 Pheromone based projects

Taking the findings of ACO one step further, [4, 8, 9] have applied pheromone based navigation to grid based graphs, representing a real 2D environment. These projects leverage the concept of stigmergy where ant-like agents find the shortest route in an environment resembling the real world. Similar to ACO, the domain generally consists of nodes connected by links. The main difference is that the resulting graph does not represent function parameters, and the nodes are spaced equidistantly. Agents do not traverse the links in iterations, but do so continuously. Like with ACO, the goal is to establish a pheromone trail on the shortest possible route between a food location and the nest entrance.

2-2-2 Modeling ants

In parallel with the research on ACO, a number of scientists [10, 11] have also studied how swarming ants behave on an individual level and consequently, modeled the ant behavior. Findings include that ants tend to follow the gradient of the pheromone distribution resulting in 'differential steering': Ants sense with their left and right antenna the concentration of pheromones on the ground and adjust their heading (and sometimes speed) accordingly. With the models formulated, the results of simulations are compared to laboratory observations with real ants. Both reports conclude that agents responding to pheromone alone does to a great extend, but not sufficiently, account for the observations from real ants.

Coherent with other studies, the differential steering is modeled probabilistic. Ants in graphs are modeled such that the probability of traversing a link (transition probability) is directly proportional to the amount of pheromone on that link. Likewise, in a continuous domain, the gradient following model is often augmented with Gaussian noise. This is considered vital for domain exploration.

The main challenge is that with the simulations, agents are stuck in cyclic trails, where they continuous reinforce their own unprofitable trail. This problem is also present in most ant simulations in a discrete environment described in Section 2-2-1 and numerous methods are used to allow agents to escape these cycles.

For controlling the agents, the model of Perna et. al [10] is particularly interesting as they provide a proof that, using their control model, the agents in a continuous domain, would display the same behavior in an artificial maze as the agents modeled according to most ACO algorithms would.

2-3 Real robots and stigmergy

If one is to demonstrate ant colony based navigation with real robots, a wide variety of marking methods are available. From vision based (glow-in-the-dark paint [12]) to temperature based (dispensing alcohol cools a surface [13], exposure to an Infrared (IR) light source heats a surface [14]), or even digital (Radio-frequency identification (RFID) tags hidden in the floor [15, 16]). Robots equipped with a dispenser or other form of actuator, can negotiate a domain while modifying the information contained within the domain (pheromone distribution). In such a way, small robots with very little computing power have successfully demonstrated that the concept of stigmergy can be used for object tracking [12, 17], domain exploration [12], spacial sorting [12] and trail exploitation [13]. Where Fujisawa et. al. [13] did successfully do exploration and exploitation based on pheromone trails, they did have to use shortcuts: the domain was very small, robots had knowledge of the nest location at all time and the robots did only dispense pheromone in a straight line between the food and nest entrance.

2-4 MSc thesis Daniel Jarne Ornia

In [4], agents traverse a square domain based on a transition probability that is modeled similar to most ACO strategies. In his experiments, Jarne Ornia models agents both with and without a directional bias that aides the exploitation part of the experiment.

As a measure of convergence of the pheromone graph (pheromone levels on all the links), the Shannon entropy of that graph is used. The quality of an established trail, can be assessed by the Shannon entropy of the pheromone graph. Pending a yet unproven conjecture, the evolution of the Shannon entropy of the pheromone graph during the experiment is a super-martingale with guaranteed convergence to a value in finite time [4, Lemma 2.2]. The convergence of the entropy consequently implies the development of a single strong foraging trail.

2-5 Discussion

Given the presented accomplished research in this chapter, to the author's knowledge, no one has accomplished applying the shortest route finding capability from Ornia [4] on a continuous domain. Equivalently, no robot swarm project (in a continuous domain), to the author's knowledge, has successfully implemented an agent strategy based solely on a response to pheromone. Even though [10,11] have presented successful ant mimicking models, their algorithms need additional logic to break out of cyclic behavior. Furthermore, these algorithms solely attempt to recreate observations that do not involve the exploitation of a food source or finding a shortest route.

With real robots, the demonstration by Fujisawa et. al. [13] of how real pheromone can be used for exploitation, comes close to exploration and exploitation based on stigmergy. It demonstrates that a marking agent can be used as artificial pheromone on real robots. Their setup however, is not similar to the agent based systems described in this chapter.

2-6 Research goals

Given the open end of the current research (Section 2-5), the goal of this thesis is to develop a pheromone based control algorithm for artificial ants and robots. The goal of the control algorithm is to have agents exploring a domain and establish a pheromone trail between the food and the agent (exploitation).

The simulation algorithm is designed in such a way that it can be implemented to real robots with a limited amount of sensing and actuating capabilities. The control algorithm can work on a fully decentralized system where all agents solely respond to pheromone and have limited awareness of their own state.

The models are tested in simulations on a scalable platform, in the sense that multiple different control algorithms, domain sizes, layouts and swarm sizes can be used. With that platform, multiple control strategies are tested such that a conclusion on the influence of the pheromone half-life time, swarm size and control algorithm can be drawn.

2-6-1 Expectations

With this thesis, I expect to create a functioning computer program that is capable of computing simulations with a large number of agents (80+) in under a few seconds. As the nature of simulations is stochastic, a great number of simulations may have to be computed in order to draw conclusions. Hence, computational efficiency is a factor.

The computer program can cope with multiple simulation strategies and store the results in a database so that the results of different simulations can be compared easily.

Both visually by creating graphs, as numerically through objective, quantifiable, performance criteria. Most important, the simulation software is open source, and thus, is publicly available.

When different control strategies are tested, a conclusion is to be drawn on the influence of all model parameters. Of special interest are the pheromone evaporation rate and the minimum number of agents required for a model to work.

The best resulting models are to be compared with the results from Jarne Ornia's thesis. This will give an insight in to what extend the findings of that thesis correlate with the findings in this work.

Ultimately, a suitable control algorithm is to be implemented on real robots in order to demonstrate a proof of concept and to analyze the correlation between the simulation and the robot demonstration.

Chapter 3

Problem description

The challenge with a control method solely based on the presence of pheromone is considered challenging, based on the findings of others. In some studies, agent based algorithms were developed for the challenge of finding a direct route (exploration and exploitation) between food and a nest entrance in a discrete domain. Others have developed agent based models in continuous domains with the aim of mimicking ant-like behavior in simulations. In this chapter, both concepts are combined and a stigmergy based, agent focused control algorithm is presented.

3-1 Experiment

Following the much used setup in literature (for example: [18, 19]) where ants are observed traversing a maze-like domain and establish a trail between the nest entrance and a single food source, the simulation is designed to do just that in an unobstructed, yet bounded, 2D continuous environment. Here, the actors in the simulation are the robots (commonly referred to as agents). These are modeled according to the characteristics of real ants.

Within the domain, a single, stationary food source is placed opposite to the nest entrance. Initially, the domain is empty. When the simulation starts, artificial ants (agents), enter the domain through the nest entrance at a certain rate. Initially, the agents explore the domain performing a random walk [20] while depositing pheromone. This transforms the random walk in a reinforced random walk [20–22] as the agents are influenced by the presence of the pheromone.

When an agent stumbles upon the food or the nest, it turns around. While the agents continue to perform the reinforced random walk, under the right conditions, a trail of pheromone should emerge between the food and nest entrance. This trail must be strong enough to 'capture' agents in the trail, such that they keep following the trail, thereby promoting efficient exploitation of the food source. If enough agents are

Table 3-1: Agent design parameters.

Name	Symbol	Description	Unit
Speed	v	Agent speed.	[mm/sec]
Length	l	Antenna length (from CR to sensor)	[mm]
Offset	d	Distance from CR to actuator location.	[mm]
Sensor offset	Γ	Angle between agent center line and sensor location from the CR	[°]

captured in a profitable trail (direct route between the food and the nest entrance), a strong and dominant foraging trail emerges.

Time

The simulation is performed in discrete time with a constant interval and a predefined number of steps N.

Definition 3.1. *Every iteration in the simulation resembles an elapse of time of δ_t seconds. With N steps, the simulation time t_{sim} at step k is*

$$t_{sim}(k) = k \cdot \delta_t, \qquad k \in N \tag{3-1}$$

3-2 Agents

The agents are modeled according to a possible ant inspired robotic layout. An overview is presented in Figure 3-1. To keep the model as simple as possible, just one pheromone sensor per side is used at a distance of l mm from the center of rotation (CR) at an offset of Γ degrees from the longitudinal axis. Aft of the CR is the actuator (pheromone dispenser) at a distance d mm from the CR. The actuator cannot be placed in the vicinity of the sensors as the sensors may then be influenced by its own pheromone additions. These parameters are to be optimized in experiments or modeled according to the morphology of real ants, summarized in Table 3-1.

Definition 3.2. *The maximum number of agents is denoted by M. The set $\mathcal{A}$ contains all $\bar{M}(k) \leq M$ active agents a indexed by m, at step k*

$$\mathcal{A}(k) = \{a_m\}_{m \in \bar{M}(k)} \tag{3-2}$$

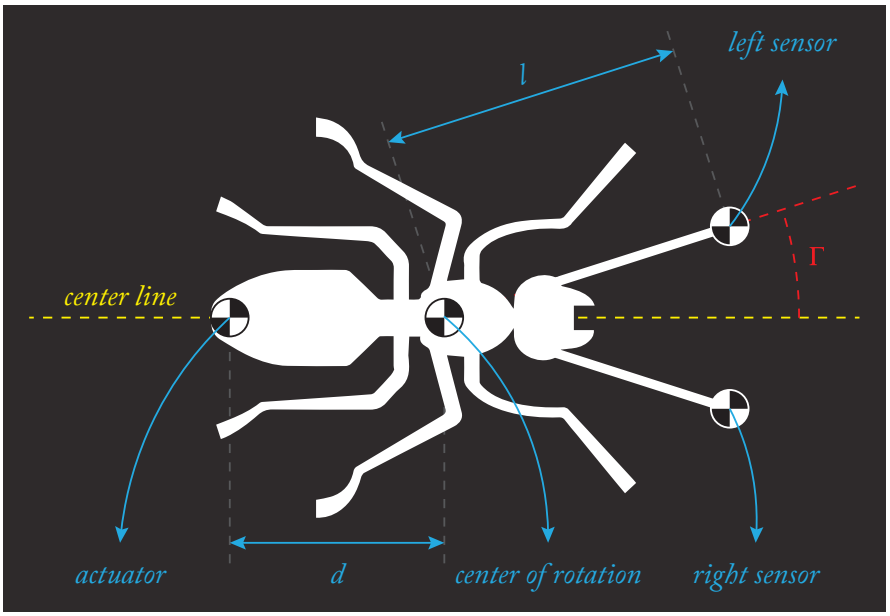


Figure 3-1: Representation of a continuous domain agent with the CR, antenna length, antenna offset and actuator offset.

3-2-1 Agent dynamics

For representing the motion of the agent, a modified version of the kinematic bicycle model [23, Eq. 1a-1e] is used. The agent heading update is slightly modified compared to the original model: the rate of change of the heading is a function of the pheromone sensed. A representation of all the relevant parameters for this model is given in Figure 3-3.

Definition 3.3. *Each agent is indexed by $m \in \bar{M}(k)$. The velocity, v, of the agents is defined in millimeters per second. In discrete steps, each agent first rotates about its CR prior to stepping forward. With an elapsed time of δ_t and measuring the heading θ from the X-axis as counter-clockwise positive, the agent position is modeled as*

$$\omega_m(k) = f(\vec{\tau}_m(k)) \tag{3-3a}$$
$$\theta_m(k+1) = \theta_m(k) + \omega_m(k)\delta_t \tag{3-3b}$$
$$x_m(k+1) = x_m(k) + v\delta_t \cdot \cos\left(\theta_m(k) + \omega_m(k)\delta_t\right) \tag{3-3c}$$
$$y_m(k+1) = y_m(k) + v\delta_t \cdot \sin\left(\theta_m(k) + \omega_m(k)\delta_t\right) \tag{3-3d}$$
$$t_m(k+1) = t_m(k) + \delta_t \tag{3-3e}$$

The perceived pheromone concentration under the left and right sensor is denoted as $\vec{\tau} = [\tau_L, \tau_R]^\top$. The angular velocity is ω and is controlled by a function of the the perceived pheromones: $f(\vec{\tau}_m(k))$. Notice that every agent internally keeps track of the elapsed time.

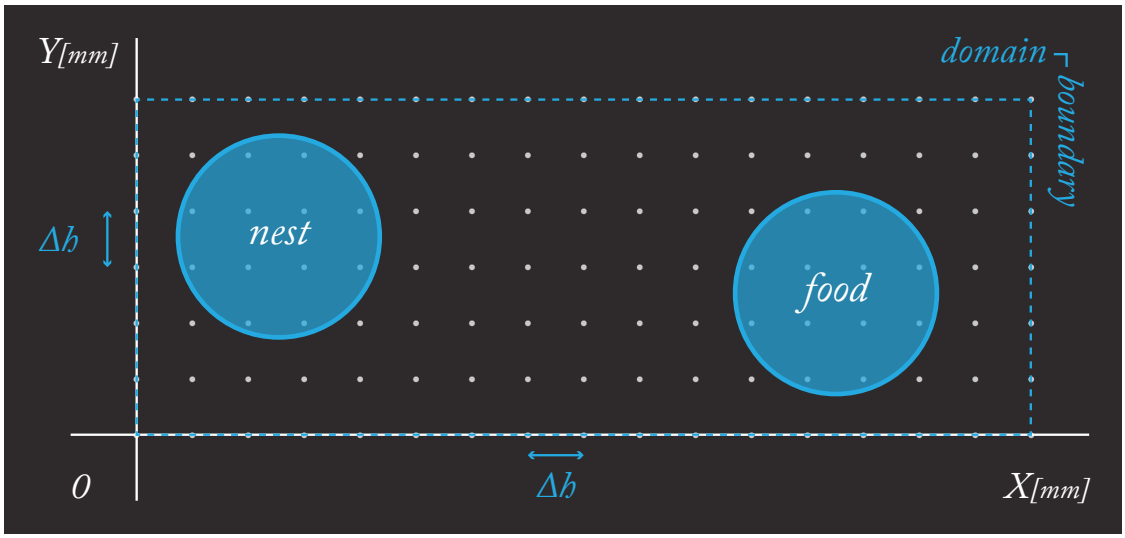


Figure 3-2: Illustration of the important properties of the domain.

Table 3-2: Overview of the domain properties.

symbol	property	description
X, Y	size [mm]	Domain height and width.
Δh	pitch [mm]	Spacing between grid points.
R	radius [mm]	Nest or Food radius.
x_f, y_f	location [mm]	Location of the center of the food w.r.t. the origin.
x_n, y_n	location [mm]	Location of the center of the nest entrance w.r.t. the origin.
x_m, y_m	location [mm]	Location of the CR of an agent.

3-3 Domain

The domain of the simulation is a rectangular 2D space, measured in millimeters. Pheromone is sampled at an equidistant spaced grid. The physical properties of the domain are presented in Table 3-2 and illustrated in Figure 3-2. Within the domain, there is a nest entrance and food location. Both are circular regions with radius R.

Definition 3.4. *The domain $\mathcal{D}$ is defined as the tuple set of coordinates that are within the domain boundaries (X, Y)*

$$\mathcal{D} = \{(x, y) \; \forall \; 0 \leq x \leq X \wedge 0 \leq y \leq Y\}, \qquad (x, y) \in \mathbb{R}^2 \tag{3-4}$$

Definition 3.5. *The set $\mathcal{I}$, is a set of equidistantly spaced nodes, indexed by tuple pairs of non-negative natural numbers (i, j) (X, Y) and is defined as*

$$\mathcal{I} = \{(i, j) \; \forall \; i\Delta h \leq X \wedge j\Delta h \leq Y\}, \qquad (i, j) \in \mathbb{Z}^{*2} \tag{3-5}$$

A graphical representation of the domain is shown in Figure 3-3. The position of an agent is defined, with $\vec{i}, \vec{j}$ the unit vectors in X and Y direction respectively, as presented

in Equation (3-6). The agent's heading is defined as counter-clockwise positive with respect to the X-axis, as shown in Figure 3-3.

Definition 3.6. *The position of an agent, identified by m, in domain coordinates is defined as*

$$\vec{\Phi}_m = \vec{i}x_m + \vec{j}y_m, \qquad (x_m, y_m) \in \mathcal{D} \tag{3-6}$$

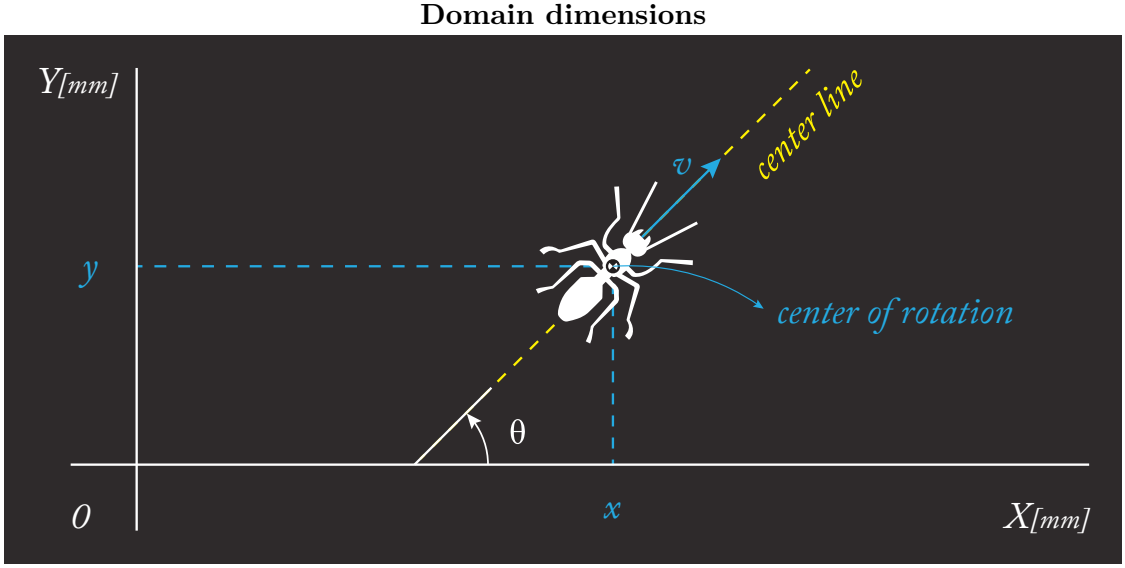


Figure 3-3: Representation of the framework for the 2D ant inspired robot simulation.

3-4 Efficiency assessment

The result of the simulation depends on multiple components: agent control model, simulation rules and the domain layout. In order to evaluate the results of different simulations, an efficiency score is presented. Defined as the amount of agents making a successful return trip from the nest entrance to the food and back again, multiplied with the minimum distance that trip takes, divided by the total distance all the agents have covered. With n agents, the score of the simulation with N steps is:

Definition 3.7. *The success of a simulation is assessed by an efficiency score, based on the amount of agents making a successful roundtrip between the nest entrance to the food, #returns, and the minimum distance that trip should take. The score is weighted by the total distance all agents have covered during the entire simulation.*

$$\Upsilon = \frac{2\sqrt{\left\| \begin{bmatrix} x_{nest} - x_{food} \\ y_{nest} - y_{food} \end{bmatrix} \right\|_2^2} \cdot \#returns}{\sum_{k=1}^{N} \sum_{m \in \bar{M}(k)} \sqrt{\|\vec{\Phi}_m(k) - \vec{\Phi}_m(k-1)\|_2^2}} \tag{3-7}$$

Notice that the efficiency score is dependent on the design parameters of the simulation and a fair amount of randomness.

One may notice that theoretically, an efficiency score greater than 1 is possible. The net minimum distance between the food and the nest is to be compensated for the radius of the nest. This difference is purely cosmetic.

3-5 Pheromone map

As the agents traverse a virtual domain, the concentration and distribution of pheromone within that domain is virtual as well. At every step in the simulation, the program is to query all agents for their position and pheromone quantity to be deposited. In the simulation, the quantity and location of pheromone on the map is represented by a graph of located weights.

Definition 3.8. *A weighted graph $\mathcal{G}(k) = (\mathcal{N}, \mathcal{W}(k))$ consists of a set of nodes $\mathcal{N}$ (pheromone locations $n_{i,j}$) and a set of weights $\mathcal{W}(k)$ (pheromone quantity $w_{i,j}$)*

$$\mathcal{N} = \{n_{i,j} \ \forall (i,j) \in \mathcal{I}\} \tag{3-8}$$

$$\mathcal{W}(k) = \{w_{i,j}(k) \ \forall (i,j) \in \mathcal{I}\} \tag{3-9}$$

3-5-1 Pheromone

The pheromone itself (quantity on the graph is represented by the weights $w_{i,j}$) is a volatile compound. Once it is deposited, the pheromone evaporates over time. Generally, the evaporation is modeled as exponential decay with a (continuous time) decay constant λ.

$$\frac{dw}{dt} = -\lambda w \leftrightarrow w(t) = w_0 e^{-\lambda t}, \qquad \text{s.t. } w(t=0) = w_0 \tag{3-10}$$

The half-life time $t_{\frac{1}{2}}$ of such dynamics reads: $t_{\frac{1}{2}} = \frac{\ln(2)}{\lambda}$. For real ants, this half-life time varies greatly among the species[1]: from ample minutes to multiple days [24, 25]. The half-life time of the pheromone in the simulation is a design parameter.

In a discrete time scenario, the evaporation rule from Equation (3-10) is transformed into:

$$w(k+1) = \rho w(k) \tag{3-11}$$

Here, the pheromone quantity w at the next simulation step is equal to the current concentration times the evaporation rate ρ. The relation between the evaporation rate and the half-life time depends in this case on the discretization of time: the time spanned per simulation step.

[1] Also, radiation from the sun, substrate material and temperature influence the half-life time

Proposition 3.1. *In order to preserve the half-life time characteristics of a pheromone throughout different simulations with different time steps, the evaporation of pheromone with an evaporation rate ρ $[s^{-1}]$ is based on a time step of 1 second. For an arbitrary large time across different simulations, the decay of pheromone is modeled as*

$$w_{i,j}(k+1) = \rho^{\delta_t} w_{i,j}(k) \tag{3-12}$$

Proof. Using Definition 3.1, by the definition of the half-life time, the initial quantity of pheromone has decayed to half that value after $k = t_{\frac{1}{2}}$ steps of 1 second ($\rho^{t_{\frac{1}{2}}} = \frac{1}{2}$), or equivalently: after $k = t_{\frac{1}{2}}/\delta_t$ steps of arbitrarily large step size (assuming k is a natural number)

$$\begin{aligned} w_{i,j}(k = \frac{t_{\frac{1}{2}}}{\delta_t}) &= \rho^{\delta_t} w_{i,j}(k-1) = (\rho^{t_{\frac{1}{2}}})^2 w_{i,j}(k-2) \\ &= w_{i,j}(0) \left(\rho^{\delta_t}\right)^{t_{\frac{1}{2}}/\delta_t} \\ &= w_{i,j}(0)\rho^{t_{\frac{1}{2}}} = \frac{w_{i,j}(0)}{2} \end{aligned}$$

□

3-5-2 Pheromone deposit distribution

The action of depositing pheromone on the domain is the equivalent of adding a quantity to the graph. This addition is according to a 2D Gaussian distribution, centered around the agent's actuator location as suggested in [11] (and others). This distribution, for example, can be thought of as representing a liquid (pheromone) being dispensed such as a robot would do [13,26]. When the simulation time steps are small enough, an agent dispensing pheromone according to such a distribution, while moving, should leave a steady, uninterrupted, line of pheromone on the domain.

Definition 3.9. *The euclidean distance between a node $n_{i,j}$ in the graph and a location $[x, y]^\top$ is defined as:*

$$r(n_{i,j}, x, y) = \sqrt{\left\| \begin{bmatrix} i\Delta h \\ j\Delta h \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \right\|_2^2} \tag{3-13}$$

Definition 3.10. *The pheromone addition is modeled with a Gaussian distribution centered around $[\mu_x, \mu_y]^\top$, with covariance $\sigma^2 = \sigma_x^2 = \sigma_y^2$. This Gaussian function $h(r, \sigma^2)$ as a function of the distance r and the covariance σ^2 from the deposit location ($r \equiv r(n_{i,j}, x, y)$) is given as*

$$h(r, \sigma^2) = e^{-\frac{r^2}{2\sigma^2}} \tag{3-14}$$

Agents add pheromone to the graph, according to Equation (3-14) at the location of the actuator. This Gaussian function is defined and non-zero for the whole domain, but practically zero everywhere except for the region in the vicinity of the deposit location. In order to save computing time, only the weights of the graph are updated where the Gaussian $h(r, \sigma^2) \geq 10^{-\iota}$ where ι represents the significant digits

Proposition 3.2. *The radius of influence around the pheromone graph is computed as the distance from the center where the Gaussian function from Equation* (3-14)*, truncated to ι digits is zero*

$$\bar{r} = \sqrt{-2\sigma^2 \ln 10^{-\iota}}, \qquad \iota \in \mathbb{Z}^* \tag{3-15}$$

Proof. The proof is straightforward, as filling in the numbers yields:

$$h(\bar{r}, \sigma^2) = \exp -\frac{\sqrt{-2\sigma^2 \ln 10^{-\iota}}^2}{2\sigma^2} = e^{\ln 10^{-\iota}} = 10^{-\iota}$$

□

Definition 3.11. *When an agent adds pheromone to the graph at location $[x, y]^\top$ the pheromone addition Δw to the graph is calculated as*

$$\Delta w_{i,j} = p \qquad s.t. \begin{cases} p = h(r(n_{i,j}, x, y), \sigma), & \forall\, r \leq \bar{r} \\ p = 0 & otherwise \end{cases} \tag{3-16}$$

3-5-3 Evolution of the graph

Each simulation step, all active agents add an amount of pheromone to the graph. The quantity per agent $q_m(k)$, does not have to be constant and may depend on the state of the agent. Each step, all active agents modify the graph by adding pheromone. After all agents update the graph, the weights are updated with evaporation.

Definition 3.12. *The evolution of the set of weights $\mathcal{W}(k)$ throughout the simulation is influenced by both the active agents (reinforcement) and evaporation. By combining Proposition 3.1 and Definition 3.11 the evolution of the graph is modeled as*

$$w_{i,j}(k+1) = \rho^{\delta_t} \left(w_{i,j}(k) + \underbrace{\sum_{m \in \bar{M}(k)} q_m(k) \Delta w_{i,j}}_{reinforcement} \right), \quad \forall w_{i,j} \in \mathcal{W} \tag{3-17}$$

3-5-4 Pheromone sampling and perception

Every agent has two pheromone sensors, as mentioned in Section 3-2. The perception of the pheromone is not necessarily linear or a 1 to 1 mapping. In line with the definition of the graph, there is no pheromone outside the boundaries of the domain.

Definition 3.13. *The mapping 'z = round(s)' from a continuous number s to its representation as a non-negative natural number z, ($\mathcal{R} \to \mathcal{Z}^*$), using the floor function is defined as:*

$$z = \text{round}(s) = \max(\lfloor s + \frac{1}{2} \rfloor, 0)$$

Definition 3.14. *The pheromone concentration is sampled at the node closest to the sample location. The mapping operator* grid $:= \mathbb{R} \mapsto \mathbb{Z}^*$ *from a location in the continuous domain $(x, y) \mapsto (i, j)$ to a node index in discrete representation is defined as*

$$(i, j) = \text{grid}(x, y) = \left(\text{round}\left(\frac{x}{\Delta h}\right), \text{round}\left(\frac{y}{\Delta h}\right)\right), \ (x, y) \in \mathcal{D} \tag{3-18}$$

Agents sense the pheromone on the domain. In the simulation, the weights of the graph located closest to the agent's sensor is sensed as pheromone level. The perception, is not necessary a 1-to-1 mapping.

Definition 3.15. *Let $\tau(x, y)$ be the perceived amount of pheromone at a location in the domain. Then, the agent's perceived amount of pheromone is a result of the perception function g*

$$\tau(x, y) = g(w_{i,j}), \qquad \exists!(i, j) = \text{grid}(x, y) \tag{3-19}$$

Definition 3.16. *The perception of pheromone is 0 by definition when the sense location is outside the domain boundary*

$$\tau(x, y) = 0, \ \forall (x, y) \notin \mathcal{D} \tag{3-20}$$

3-6 Control algorithm

Given the agent dynamics, it is apparent that the only control input is the pheromone based function in Equation (3-3a). This input $f(\vec{\tau}(k))$ is solely based on the perception of the sensor readings. The challenge is to design a pheromone based input function that exhibits the reinforced random walk behavior, and allows agents to follow a pheromone trail where present. Two controllers are presented here.

3-6-1 Basic control model

The most simple model, based on positive reinforcement, is go left when $\tau_L > \tau_R$, and right otherwise: $f_s(\vec{\tau}(k)) = \alpha(\tau_L(k) - \tau_R(k))$, with a simple, constant, feed forward gain α. To achieve efficient ant-like domain exploration, the random part of the reinforced random walk can be ascribed to sensor. With two additive identically distributed independent white noise sources on the sensors, Equation (3-3a) is written as:

$$f_s(\vec{\tau}(k)) = \alpha \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} \tau_L(k) + \epsilon_l(k) \\ \tau_R(k) + \epsilon_r(k) \end{bmatrix} \quad \text{s.t.} \quad \begin{cases} \epsilon_l \sim \mathcal{N}(0, \sigma^2) \\ \epsilon_r \sim \mathcal{N}(0, \sigma^2) \end{cases} \tag{3-21}$$

Equation (3-21) can be re-written in a form with just a single source of noise that shares the same statistical properties:

$$\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} \tau_L(k) + \epsilon_l(k) \\ \tau_R(k) + \epsilon_r(k) \end{bmatrix} \equiv \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} \tau_L(k) \\ \tau_R(k) \end{bmatrix} + \epsilon(k) \;\; \text{s.t.} \;\; \begin{cases} \epsilon_l, \epsilon_r \sim \mathcal{N}(0, \sigma^2) \\ \epsilon \sim \mathcal{N}(0, 2\sigma^2) \end{cases} \tag{3-22}$$

Yielding:

$$f_s(\vec{\tau}(k)) = \alpha \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} \tau_L(k) \\ \tau_R(k) \\ \epsilon(k) \end{bmatrix} \quad \text{s.t. } \epsilon \sim \mathcal{N}(0, 2\sigma^2) \tag{3-23}$$

Equation (3-23) can also be written as:

$$f_s(\vec{\tau}(k)) = \epsilon(k) \qquad \text{s.t. } \epsilon(k) \sim \mathcal{N}(\alpha(\tau_L(k) - \tau_R(k)), 2\alpha^2\sigma^2)$$

Definition 3.17. *The 'simple' control algorithm for controlling the heading of the agent is a random process. It is modeled as a distribution centered around the (scalar) difference in perceived pheromone concentrations and reads*

$$f_s(\vec{\tau}) = \epsilon(k) \qquad s.t. \; \epsilon(k) \sim \mathcal{N}(\alpha(\tau_L(k) - \tau_R(k)), 2\alpha^2\sigma^2) \tag{3-24}$$

Sensor activation

Although this 'simple' model fulfills the characteristics of a reinforced random walk, there are caveats. In the first place, the output is not bounded from above. If the difference in pheromone concentration is high, the output and resulting angular speed ω may be much higher than desirable. Second, the magnitude of the output is directly related to the amount of pheromones the agents deposit. If, for example, the agent deposition rate is updated or the swarm size is enlarged, the steering gain may have α to be adjusted accordingly.

In order to remedy these problems, the way agents perceive the pheromone concentrations can be altered. With modeling ants, [26–28], various researchers successfully used a sigmoidal pheromone perception function as an artificial way to model sensor saturation. The additional benefit is that this nonlinear perception can help avoid traffic agglomeration or help an agent escape an unprofitable trail. In such a case, Equation (3-19) $g(w_{i,j})$ would read:

$$\tau = g(w_{i,j}) = \frac{1}{1 + e^{-w_{i,j}}} \tag{3-25}$$

However, in line with the theory of no free lunch [29], there is a caveat. Without the linear relation between actual present and perceived pheromone, when the agent deposition rate is adjusted in between experiments, it may prove very difficult to adjust the steering gain accordingly.

3-6-2 A Weber's law model

As an alternative to the basic control model, an agent model that follows Weber's law [30]: the perception of the pheromone difference is relative to the total sum of sensed pheromone.

Definition 3.18. *Following Perna et. al. [10], who have successfully designed such a control model for artificial ants, the second control model is called 'Weber' and defined as*

$$f_w(\vec{\tau}) = \alpha \left(\frac{\tau_L(k) - \tau_R(k) + \epsilon_1(k)}{\tau_L(k) + \tau_R(k) + \kappa} + \epsilon_2(k) \right) \delta_t \quad s.t. \quad \begin{cases} \epsilon_1 \sim \mathcal{N}(0,\, \sigma_1^2) \\ \epsilon_2 \sim \mathcal{N}(0,\, \sigma_2^2) \\ \kappa > 0 \end{cases} \tag{3-26}$$

Here, the gain α, and the noise characteristics of ϵ are tuning parameters (for example, $\sigma_1 = 50, \sigma_2 = 15$ in [10]). One modification is the addition of a small non-negative scalar κ in order to prevent the denominator to become zero.

The major advantage of this system is that the dynamics of the agent become more or less independent of the deposition amount, compared to Equation (3-23). Hence, the use of a perception function is redundant as the denominator in Equation (3-26) acts as a form of auto regulation or sensor saturation. Since the pheromone concentration is always non-negative, neglecting the noise, the response magnitude is bounded by the choice of steering gain α.

One challenge may be that the noise term $\epsilon_1(k)$ may become dominant when the agent explores patch of the domain that has not been traversed by other ants. Hence, the pheromone levels on the graph may be very low, rendering $\epsilon_1(k)$ to be very influential and forcing the agents to have a very large, fluctuating angular velocity. To circumvent this problem, the rectifying linear perception function (often called ReLu when applied to artificial neural networks) is defined.

Definition 3.19. *The rectified linear perception function with a breakpoint at z_{br} is defined as*

$$\tau(x, y) = g(w_{i,j}) = \max(z_{br}, w_{i,j}), \qquad \forall (i, j) \in \mathcal{I} \wedge (x, y) \in \mathcal{D} \tag{3-27}$$

With this model, the observation of pheromone is usually a 1 to 1 mapping, but it provides a lower bound for the denominator term as it is now bounded from below. This inversely bounds the influence of the noise term ϵ_1.

This model contains two sources of noise. One of constant magnitude and the other dependent on the pheromone sum. With the latter, agents behave less random in the presence of high amounts of pheromone. This is in accordance with many research papers [5], including the MSc thesis of Jarne Ornia [4], where agents initially perform a random walk during an exploration phase that transforms into an exploitation phase (less randomness).

3-7 Challenges with stigmergy

The concept of stigmergy prescribes that agents respond to the pheromone and nothing else. Yet, research with other agent models such as [10, 11, 31], indicate that simple agents are prone to be trapped in sub optimal routes: cyclic routes or routes along the domain boundary. In order to minimize the decrease of efficiency in exploitation from these phenomena, three possible counter-measures are presented.

3-7-1 Pheromone deposition

Generally, a slow decaying pheromone allows for efficient exploitation. Once a trail is formed, it does not readily disappear. Agents follow and reinforce the trail thereby making it even stronger which helps capturing exploring agents in the exploitation trail. It is possible that the swarm size can be small as it requires few agents traversing the trail to maintain it. In other words, with slow evaporation of pheromone, there is less reinforcement required for the reinforcement to be stronger than the evaporation.

The downside of the slow evaporation method, is that agents are also prone to follow trails that do not directly link the nest and the food source (unprofitable trails). To prevent unprofitable trails from persisting through the simulation, the pheromone deposition $\Delta\tau$ for each agent is time and state based.

Definition 3.20. *A time dependent pheromone addition function for Definition 3.12, strictly decreasing with with time, is based on the agent internal timer t_m and defined as*

$$q_m(k) = 1 - q_0 e^{-\gamma t_m(k)} \tag{3-28}$$

Here, q_0 is the default deposition quantity for all agents, γ is the deposition decay rate (not the same as in Equation (3-10)). Both q_0 and γ are design parameters.

With this time dependent addition, the longer the agent explores the domain without encountering food or the nest, the less pheromone it deposits. The internal timer is reset each time an agent successfully makes its way to a food or nest source. This way, agents that are on a profitable trail will drop more pheromone than other agents. Additionally, by making q_0 state dependent, the agent can be modeled such that the pheromone deposition quantity is more when it is returning to the nest, compared to searching food.

3-7-2 Noise

By changing the characteristics of the noise distribution, the agent exploration behavior can be modified, and to a lesser extend, the exploitation behavior. Vela-Perez et. al. [11] mentioned a run-and-tumble characteristic observed with real ants. Similarly, Couzin and Franks [26] mention ants have a tendency to perform u-turns. They noticed that after some time of exploring, ants can suddenly move in a completely different direction. It is thought that this behavior can help escaping unprofitable trails although the setting in which this was studied, only allowed for exploration and not exploitation.

Run-and-tumble behavior can be induced by using a special form of noise: a special time based noise series. By summing Gaussian noise with a random telegraph time series [32]. Here, the telegraph time series $\varepsilon(t_m(k), T_s(\beta))$ is modeled as a time series with two discrete states $(S_{\text{low}}, S_{\text{high}}) = (-0.5, 0.5)$. The time between two state changes (sojourn time) $T_s[s]$ is modeled as an exponentially distributed stochastic process.

Definition 3.21. *The time based, augmented telegraphic noise series ς, is a series of random noise defined as the sum of a telegraph noise series ε and a normal distributed noise*

$$\varsigma(t_m(k), T_s(\beta), \sigma^2) \equiv \varepsilon(t_m(k), T_s(\beta)) + \epsilon(k) \text{ with } \begin{cases} T_s \sim Exp(\frac{1}{\beta}) \\ \epsilon \sim \mathcal{N}(0,\, \sigma^2) \end{cases} \tag{3-29}$$

The tuning parameter β, determines the length of the sojourn time. When set just right, such that the state of the random telegraph process changes frequently compared to the simulation time step, the agent remains, relatively unaffected by the telegraph process. However, due to the characteristics of the exponential distribution: with a small possibility, the sojourn time is large, spanning multiple time steps, forcing the agent in a random orientation. This mimics the run-and-tumble behavior [11] or tendency to make U-turns [33].

Definition 3.22. *The 'simple' control model augmented with a telegraph noise series, with a feedforward gain for the rotation sensitivity α_1 and feedforward gain for the noise sensitivity α_2, is modeled as*

$$f_w(\vec{\tau}) = \alpha_1 \left(\tau_L(k) - \tau_R(k) + \alpha_2 \varsigma(t_m(k), T_s(\beta), \sigma^2)\right) \tag{3-30}$$

3-7-3 Navigational memory

A third method that may be promising is by ascribing a sense of directionality to the agents. Multiple researchers have modeled artificial ants with directional awareness [4,26,34], leading to better exploitation behavior: searching the nest is still exploration, once trails establish and food has been found, the agent is biased to walk in the direction of the nest. This overrides the pheromone response.

This can be modeled similar to [4], where exploring agents memorize the number of vertical and horizontal steps. By keeping track of the vertical and horizontal distance covered while the agent is looking for food, an accurate estimate of the direction of the nest entrance is computed when the agent finds the food.

Definition 3.23. *Consider the agent based 'nest-vector' $\vec{V}_m$ defined at the instant the agent left the nest at step k_1, the nest vector for an agent is defined as*

$$\vec{V}(k)_m = \sum_{k=k_1}^{k} \left(\begin{bmatrix} x_m(k-1) - x_m(k) \\ y_m(k-1) - y_m(k) \end{bmatrix} \right) = \begin{bmatrix} x_m(k_1) - x_m(k) \\ y_m(k_1) - y_m(k) \end{bmatrix} \tag{3-31}$$

Then, the required agent heading to the nest is $\angle\vec{V}_m$ (degrees).

Definition 3.24. *When an agent is nestbound, consider agent's error angle in degrees: $\hat{\theta}_m$ which is the angle difference between the agent heading and the direct heading to the nest*

$$\hat{\theta}_m = \angle\vec{V}_m - \theta_m(k), \qquad -180 < \hat{\theta}_m \leq 180 \tag{3-32}$$

Now, the required (desired) angular velocity to face the nest at the next rotation update, is $\omega_m^{\text{des}}(k) = \frac{\hat{\theta}_m}{\delta_t}$.

A suitable control law for a nest-bound agent as modeled in Equation (3-3) at step k is for a nestbound agent to be increasingly biased to steer towards the nest, with a maximum 'override coefficient' η.

Definition 3.25. *An alternative control law to Equation* (3-3a) *is called 'directional override'. It provides a bias for the agent to face the nest.*

$$\omega_m(k) = f(\vec{\tau}_m(k))(1 - g_m(k)) + g_m(k)\omega_m^{des}(k) \tag{3-33}$$

$$g_m(k) = \min(\eta \frac{t_m(k)}{t_{max}}, \eta) \quad s.t.\ 0 \leq \eta \leq 1 \tag{3-34}$$

Here, $t_m(k)$ is based on the agent internal timer. η is the maximal 'override' fraction. If $\eta = 1$, the agent heads directly to the nest after t_{max} seconds after it became nestbound. On the other hand, if η is close to zero, the agent only has a slight bias to steer to the nest and remains capable of following pheromone trails. A visualization of the override influence is shown in Figure 3-4.

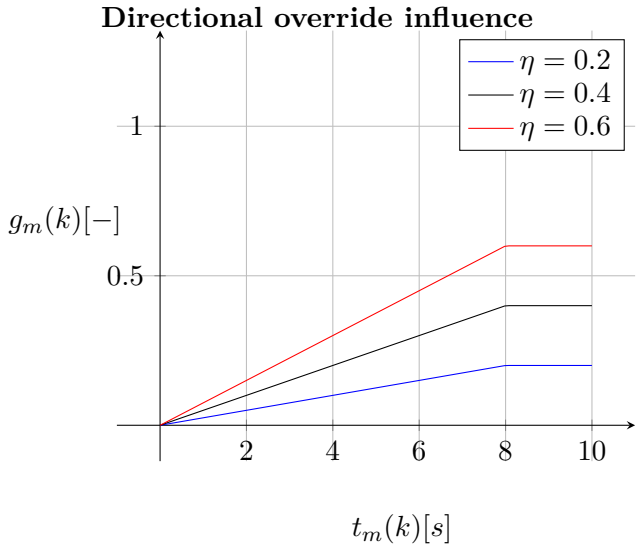


Figure 3-4: Magnitude of the override function for different values of η (Equation (3-34)) with $t_{\text{max}} = 0.8$.

3-8 Discrete navigation

The work in this thesis is inspired by the results from [4]. The simulations and theories all take place in a discrete environment. The domain is a square grid consisting of equidistant spaced nodes. All direct neighboring nodes are connected through horizontal or vertical aligned links. Agents are represented as point, located on a vertex. An agent can move one node at a time, by traversing the connecting link.

3-8-1 Shannon entropy

The quality of the solution (length and strength of the foraging trail) is assessed by evaluating the Shannon entropy of the graph.

Definition 3.26. *With the total amount of pheromones calculated as* $T(k) = \sum_{w_{i,j} \in \mathcal{W}(k))} w_{i,j}$ *the Shannon entropy is defined as*

$$H(\mathcal{W}(k)) = -\sum_{w_{i,j} \in \mathcal{W}} \frac{w_{i,j}}{T(k)} \log_2 \frac{w_{i,j}(k)}{T(k)} \tag{3-35}$$

Analogous to [4], the entropy of the graph as presented in Definition 3.8 is maximum when the pheromone distribution over the graph is completely uniform, e.g. at the start of the simulation with $t = 0$. The maximum value solely depends on the size of the graph:

$$H_{\max}(\mathcal{W}(k)) = \log_2(|\mathcal{L}|) = H(\mathcal{W}(k = 0)) \tag{3-36}$$

Similarly, the minimum entropy is when all the pheromone agglomerates on a single link. In such a case, the minimum entropy approaches zero: $H_{\min}(X) \approx \frac{T}{T} \log_2 \frac{T}{T} = 0$.

The Shannon entropy is used to assess the quality of a solution: the shorter the trail between the nest and the food, the smaller the entropy of the graph. In the discrete scenario, under the assumption that the evolution of the graph is a super-martingale, the pheromone graph is guaranteed to converge in finite time [4, Lemma 2.2].

Chapter 4

Computer simulations

The ideas for controlling a swarm of autonomous operating agents are put to the test in a series of simulations. Both the the control algorithms performance and the software computation time are of interest. To that extend, a detailed description of the simulation software is presented in Appendices A and B.

4-1 Goal

The simulations are performed to gain an understanding in to what extend, simple sensor driven algorithms, can lead to complex swarm intelligence. Hence, the goal of the simulations is to test the capacity of a swarm of autonomous agents in general, and the influence of the pheromone characteristics.

In order to assess the effectiveness of the algorithm, the properties of the agents and the domain, a score as mentioned in Section 3-4 and expressed in Equation (3-7) is used. The challenge of tuning the domain is then formulated as finding the optimal set of parameters, denoted as set $\mathcal{S}^*$ that yields the maximum efficiency score:

$$\mathcal{S}^* = \underset{\mathcal{S}}{\operatorname{argmax}}\, \Upsilon(\mathcal{S}) \tag{4-1}$$

The set of design parameters is shown in Table 4-1. In the experiments presented in this document, the deposition quantity and its decay is fixed (set and forget).

As the actions of each agent can be regarded as a stochastic process, the outcome of the simulation is not deterministic. Therefore, one can only assess the effectiveness of the control algorithm based on a large number of experiment repetitions.

Table 4-1: Simulation design parameters

Symbol	Parameter	Comment
$f(\vec{\tau}_m(k))$	Control function	Equation (3-3a)
q_0	Base deposit quantity	Equation (3-28)
$q(k)$	Deposition decay	Equation (3-28)
σ^2	Deposition covariance	Equation (3-14)
$g(w_{i,j})$	Perception function	Equation (3-19)
η	Directional override	Equation (3-34)

4-2 Agents

Most agent properties are described in Section 3-2. Next to the general properties, each agent is assigned a state. This state indicates its position and orientation in the domain, and the role of the agent. The state variables are shown in Table 4-2

Table 4-2: Agent state variables.

Name	Symbol	Description	Unit
ID	-	Unique identifier number.	[-]
Position	$\vec{\Phi}$	Position vector of the CR w.r.t. the domain origin.	[mm, mm]
Orientation	θ	Agent orientation w.r.t. the x-axis, counter-clockwise positive.	[°]
Rotation speed	ω	Rotation speed: $d\theta/dt$	$[°/sec]$
Foodbound	-	Boolean indicating agent is looking for food.	[-]
Nestbound	-	Boolean, opposite of Foodbound.	[-]
Out of bound	-	Boolean indicating when the agent is at the border of the domain.	[-]
Timer	$t(k)$	Time since last major event.	[sec]
Deposit quantity	$q(k)$	Timer based pheromone deposition quantity.	[-]

4-2-1 Agent sub-states

Depending on the position of the agent or a sequence of past events, the agent has multiple, mutual exclusive sub-states, dictating the role of the agent in the simulation. Effector cues are defined for each sub-state: the activity an agent is performing while in that specific sub-state. Perceptual cues are the events that lead to an agent switching to a new sub-state. The sub-states, effector and perceptual cues are displayed in Table 4-3, the interaction among the sub-states is shown in Figure 4-1.

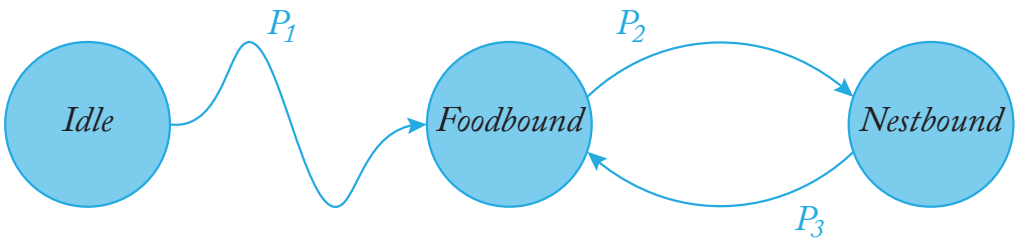


Figure 4-1: Visualization of the interaction among the sub-states, perceptions (P_1, P_2, P_3).

Table 4-3: Action algorithm for states depicted in Figure 4-1

Sub-state	Perceptual cue		Effector cue
Idle	P_1	Deploy	Wait.
Foodbound	P_2	Contact with food	Reinforced random walk.
Nestbound	P_3	Contact with nest	Reinforced random walk.

When the simulation has started, each agent starts idle at the nest until it is deployed. When the agent leaves the nest, it takes the sub-state 'foodbound' and looks for food while depositing pheromones. This behavior is called the reinforced random walk.

Events

When an agent finds food, an 'event' has happened. The agent becomes 'nestbound': the internal timer resets to zero and the agent turns around, looking for the nest and marking the environment with pheromone. Upon arrival at the nest (also an event), the agent becomes foodbound again and the cycle restarts, as illustrated in Figure 4-1.

The simulation software counts the number of these events happening, and the frequency of sub-states changing from nestbound to foodbound is called the number of returns and is used for the efficiency assessment of the simulation as explained in Section 3-4. Both events, finding the food and finding the nest, cause the agent internal timer to reset to zero. In the case an agent arrives at the nest while it is foodbound, or arrives at the food while it is nestbound, the arrival is not regarded as an event happening, the agent is simply turned around.

Agent deployment

As illustrated in Figure 4-1, at the start of the experiment, all agents are idle. An agent's deploy time is modeled according to a certain probability distribution. This is because it is unlikely that in the case of real robots, they can leave the nest entrance at the same time. Coincidentally, in experiments with real ants, the probability an ant leaves the nest is often also modeled according to a distribution such as an exponential or Gamma distribution [35] (for example [10, 36]).

All agents are deployed at the edge of the nest entrance, along the radial lines from the nest center (facing outwards). The simulation deployment time t_d is calculated according to a Gamma distribution throughout this document.

Reinforced random walk

When the agent is active, it performs the reinforced random walk. It deposits a quantity of pheromone $q_m(k)$(Equation (3-28)) on the domain while the direction of the agent is updated each step according to the specified control law (Section 3-6).

4-3 Simulation intelligence

The simulation contains three major components. First, there is the domain, the playground of the simulations. Second, there is the swarm of agents, extensively discussed in the previous section. Third, there is the simulation intelligence itself: the set of algorithms that dictate the working of the simulator. The sequence of consecutive steps is (loosely) depicted in Figure 4-2.

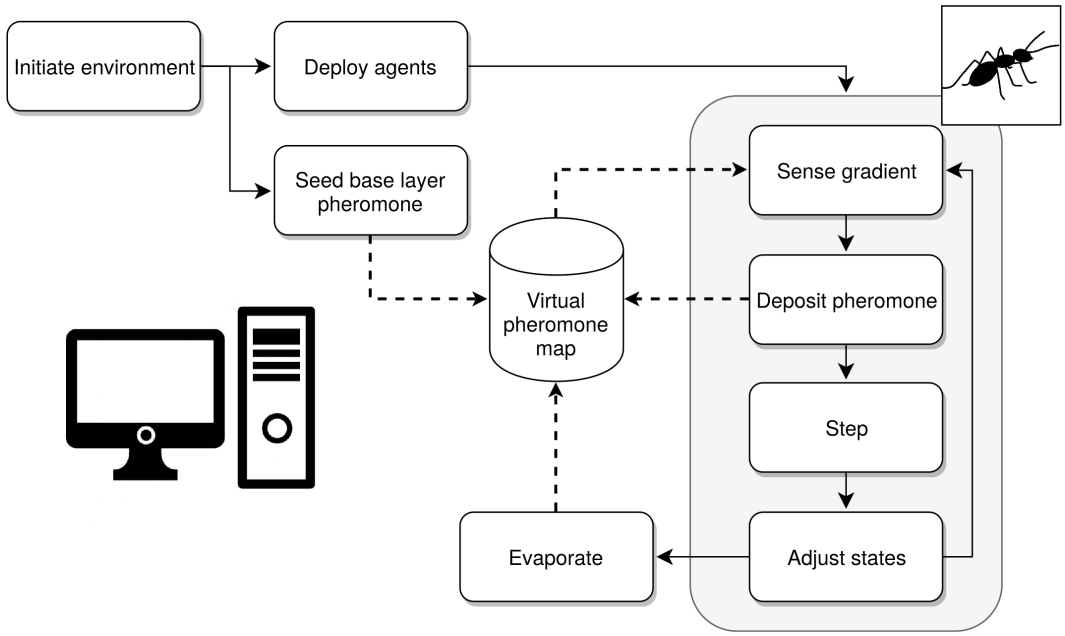


Figure 4-2: (Schematic) outline of the simulation. Striped arrows indicate interactions with the virtual pheromone map. Solid arrows represent the simulation sequence order. The shaded box indicates the actions on an agent level.

Initially, a pheromone graph is created with an initial pheromone level uniformly distributed. When the simulation starts, agents enter the domain at the nest entrance. Every time step, all active agents first sense the pheromone concentration at the left and right sensor location. Then, every agent deposits an amount of pheromone on the graph (Definition 3.12) and rotates according to the control law before making a step

forward along its centerline. Finally, the position of the agent is checked and possibly corrected for the domain properties. This last step is where the domain boundaries are enforced and the arrival at food or the nest is checked. After all agents have completed a step, the pheromone levels get an evaporation update according to Proposition 3.1 and the cycle repeats.

Domain boundary enforcement

Besides from the states that indicate the behavior of the agent, it can also be marked as being out bounds. This condition is true if any part of the agent (sensors, actuator or CR) is located outside the boundaries of the domain, and false otherwise. When the agent is out of bounds, the pheromone contribution is zero by definition: $q_m(k) = 0$. Furthermore, as per Definition 3.15, if the location of a sensor is not within the domain boundaries, that sensor observes zero pheromone regardless of the sensor activation function.

4-4 Simulation default settings

As explained before, the agents are modeled according to real ants and robots. One such robot available for real experiments is the Elisa3 [37], and where possible, the parameters are chosen to represent the Elisa3. Unless stated otherwise, the parameters presented in Table 4-4 are the default for all simulations. In Appendix C, a full parameter breakdown is presented per experiment. The simulation runs over 2000 steps, with a time step such that the pheromone deposition per ant resembles a line, rather than patches with a larger time step. The half-life time of both the deposited pheromone as the time dependent deposition quantity is approx. 23 seconds.

In Table C-1, there is a row with 'return factor'. This is a parameter that scales the pheromone addition per agent and is based on the agent sub-state. The idea is that when an agent finds the nest, it is likely the agent is on a profitable route. To accelerate the reinforcement of that that route, the deposit quantity $q_m(k)$ is multiplied with the return factor when it is nestbound. For all experiments, a return factor of 2 works well.

4-5 Scenario 1: simple control algorithm

In order to create a benchmark to compare all the simulation results with, the simplest algorithm is put to the test. To see if the agents benefit from directional memory (Section 3-7-3), three different control cases are considered:

1. Simulation using the simple control law without directional override.
2. Simple controller with a **small** fraction directional correction.
3. Simple controller with a **large** fraction of directional correction

Table 4-4: General simulation settings, for a full overview, see Appendix C.

Subject	Parameter	Symbol	Value	Remark
Simulation	Steps	N	2000	
	Time step	δ_t	0.3 sec	10 minutes of simulation time
	Swarm size	M	120	Maximum number of active agents
Agent	Speed	v	125 mm/sec	
	Antenna length	l	50 mm	Similar to Elisa3 size [37]
	Stinger distance	d	25 mm	
	Sensor offset	Γ	45 deg	
Domain	Pitch	Δh	10mm	
	Size		[3000,1500] mm	
	Food location		[2750,750] mm	
	Nest location		[250,750] mm	
	Nest, food radius		150 mm	

For each case, 100 repetitions of the simulation are carried out and the resulting efficiency statistics are summarized in a boxplot. In order to understand the impact of the different settings, some simulations are analyzed on an individual level.

4-5-1 Controller tuning

For the simple control model, the difficulty is balancing the agent exploration capabilities with the exploitation capabilities. It turns out, that without much noise covariance, the agents are prone to be stuck at the boundary: facing the boundary while stepping forward. In order to overcome this problem, the noise is to be increased but this impedes the capability to follow established foraging trails. The best tradeoff is using the telegraph noise as per Definition 3.22. The sigmoidal activation function Equation (3-25) was tested but the parameters proved too challenging to tune manually. The specific controller settings are summarized in Table 4-5.

Table 4-5: Settings for scenario 1, the simple control model.

Parameter	Symbol	Setting
Steering gain	α_1	0.75
Noise gain	α_2	1
Noise type	ς	Telegraph noise
Sojourn time parameter	β	10
Sensor activation	$g(w_{i,j})$	1:1 $\tau = w_{i,j}$

4-5-2 Results

The resulting efficiency scores for the three different cases are shown in Figure 4-3 and Table 4-6. A quantitative evaluation of the results is: rather poor. The case where only a modest amount of directional override is used, has a marginally better performance compared to the other two.

In depth analysis of the simulations learns that with too much override, the agents do not follow specific trails when returning to the nest as the directional override steers them away from the trail when the trail is not perfectly aligned with the agent's desired heading. Furthermore, the score is limited by the large number of agents that are 'stuck' at the boundary: when both sensors of the agent are at the boundary, the only source of steering comes from the additive noise. Noise with a high covariance, and long sojourn times in the telegraph noise series (Section 3-7-2), causes the agents to successfully break away from the boundary, this however, is at the cost of the ability to follow pheromone trails.

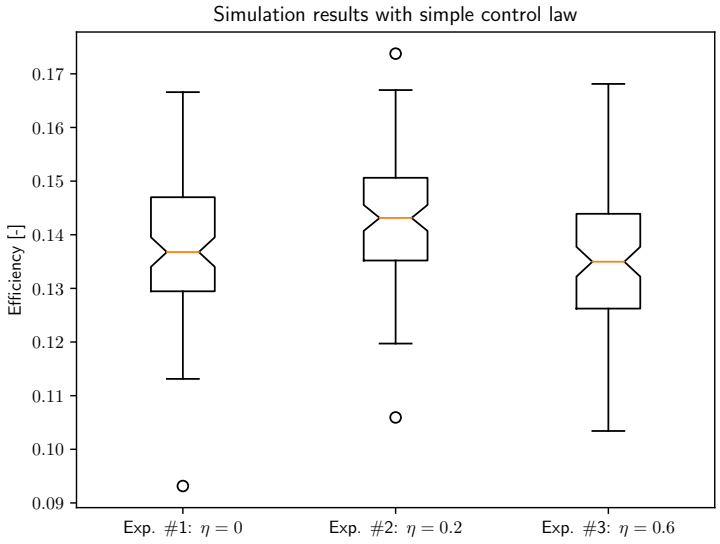


Figure 4-3: Efficiency rating for the different cases using the simple controller. The corresponding statistics are presented in Table 4-6.

Table 4-6: Efficiency rating for different amounts of directional bias (override) using the simple control algorithm. Visualized in Figure 4-3.

Efficiency (Υ)	$\eta = 0$	$\eta = 0.2$	$\eta = 0.6$
Mean	0.138	0.142	0.135
Variance	$1.7 \cdot 10^{-4}$	$1.3 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$
Min	0.093	0.106	0.103
Max	0.167	0.174	0.168

4-5-3 Individual analysis

In order to gain an understanding into what is happening at the simulations, one of the simulations from the second case (simple controller with modest amount of override) is analyzed. The development of the pheromone graph is visualized in Figure 4-4 on page 37. The simulation is one from the set with a moderate amount of override for nestbound agents: $\eta = 0.2$. The full breakdown of all the simulation parameters is shown in Appendix C, Table C-1 column header 'scenario 1'.

At the start of the simulation, some agents quickly discover the nest. But also, early in the simulation, the agglomeration of pheromone around the nest entrance causes agents to remain there while experiencing heavy fluctuations in rotational velocity. These agents, trapped in cyclic trails deposit less pheromone over time. After a little over 20 seconds ($\rho^{23} \approx 0.5$) into the experiment, they only deposit half the quantity of pheromones compared to the start. This causes the unprofitable trails to slowly evaporate while the profitable ones are continuously being reinforced. This effect is shown in Figures 4-4b and 4-4c. The pheromone around the nest is almost completely evaporated.

The telegraphic noise helps agents stuck at the boundary to join the domain again but at the cost of the ability to follow trails. Hence, agents that exploit a trail frequently lose track of it and start to create or reinforce other trails. As a result, many different pheromone trails are developed and as many evaporate and disappear (collapse).

When a trail disappears, it often happens rather abrupt: a collapse. Usually, due to the random behavior, the trails grow wider over time. At some point, none of the agents are capable of following it and abandon the trail. Abandoning the trail leads to more pheromone being deposited at the side of the trail thereby encouraging other agents to abandon the trail as well, setting off a chain reaction.

In Figure 4-5 on page 38, the number of agents successfully returning to the nest over time is shown. The graph is almost completely linear as is common in simulations where pheromone trails are omnipresent. The short intervals where the score evolution stagnates occur when a trail collapses. The linearity also indicates that the number of agents on a foraging trail remains constant.

4-6 Scenario 2: Weber's law based controller

The Weber control method should combat the deficiencies of the simple control method. The noise contribution of ϵ_1 is high during exploration (when the observed pheromone levels are low), and low when the agent exploits a trail with high levels of pheromone. Again, the same three cases as with the simple control model are considered.

4-6-1 Controller tuning

For the Weber control method, the steering feedforward gain is slightly increased, the noise is modeled according to regular white noise. A small regularization term is ap-

Table 4-7: Settings for scenario 2, the Weber control model.

Parameter	Symbol	Setting
Steering gain	α	1.25
Noise covariance 1	σ_1	0.5
Noise covariance 2	σ_2	0.1
Sensor activation	$g(w_{i,j})$	ReLu
Rectification minimum	z_{br}	0.25
Regularization term	κ	0.01

plied to the denominator of the control formula to prevent dividing by zero when both left and right sensor of an agent are out of bounds and sense zero pheromone subsequently. In order to suppress the exploration noise as mentioned in Section 3-6-2 and definition 3.19, a linear rectified perception function is used. This is because, during some time into the experiment, the areas far away from the nest may not have received any pheromone contribution while the initial pheromone presence has almost completely evaporated. When an agent observes very low values of pheromone, the numerator term in Equation (3-26) causes the noise ϵ_1 to become very large in magnitude resulting in very large direction changes. A lower bound in the pheromone perception counteracts that problem. The control settings are presented in Table 4-7.

Next to the control parameters, the hardware model is slightly changed as well. These parameters work better with this control algorithm. First, the covariance of the pheromone deposition distribution is lowered, which seems to increase the performance compared to the covariance from the first scenario. The resulting narrower trails require the sensors to be closer together. Finally, the actuator location is placed slightly forward. These modifications contribute to the trail exploitation capacity of the swarm. The full simulation setting are shown in Table C-1.

4-6-2 Results

The statistics of the resulting efficiency for all tests are shown in Table 4-6 and fig. 4-6. The performance of this second control algorithm is obviously better than the simple control algorithm. Like in the previous case, a directional bias for ants returning from the nest helps the colony to converge to efficient foraging trails. A major improvement, is that when agents have one or both sensor outside the domain boundary, they sense zero pheromone. This control method makes the agent very susceptible to the noise which results in the agents randomly steering away from the boundary. As a result, no agent is trapped at the boundary for long.

In order to investigate the rate of convergence to a single pheromone trail, the achieved efficiency for the last 200 steps in the experiments is shown versus the global efficiency of the experiment in Figure 4-7 on page 39. One can see that the efficiency late in the experiment is lower in some cases for the control algorithm without override, which may be a sign of late collapsing of the primary foraging trails.

Table 4-8: Efficiency rating for different amounts of directional bias (override) using the Weber's law inspired control algorithm. Visualized in Figure 4-6.

Efficiency (Υ)	$\eta = 0$	$\eta = 0.2$	$\eta = 0.3$	$\eta = 0.6$
Mean	0.633	0.674	0.749	0.779
Variance	$6.136 \cdot 10^{-2}$	$6.469 \cdot 10^{-2}$	$6.706 \cdot 10^{-2}$	$5.937 \cdot 10^{-2}$
Min	0.15	0.163	0.121	0.126
Max	1.052	1.038	1.072	1.078

On the other hand, the samples located above the centerline corresponding to $x = y$, especially in the upper left quadrant, indicate that a very efficient foraging trail has established late in the simulation.

4-6-3 Individual analysis

To gain a solid understanding of the events that lead to convergence, or the absence of it, three different cases are analyzed: the first is the control method without directional bias, with a poor efficiency. The second is a very exemplary result for the average performance. The last case is one of the best results observed, using a high amount of override.

It turns out that the entropy is representative for the quality of the trail in converged cases. So, instead of the number of returning agents over time, the evolution of the entropy of the pheromone graph is presented.

Example of slow convergence to a sub-optimal trail

As a first example, a simulation that has a low score is analyzed. There is no directional bias implemented. The resulting score is $\Upsilon = 0.191$. One can see in Figure 4-8 that some unprofitable trails form at the start of the simulation (Figure 4-8a). This is because some ants tend to follow the fresh pheromone deployed by other agents. Once this effect dies out due to the deposition decay, the unprofitable trails collapse.

Eventually, a strong foraging trail develops. The problem with this trail is, that agents eventually interact with the boundary and a portion of the agents randomly turns around. This typical behavior is counteracted by the directional bias as discussed in the next example.

The development of the entropy of the pheromone graph is presented in Figure 4-9. A low entropy corresponds with strong and short (efficient) pheromone trails. The entropy of the graph rises when a trail collapses and the agents scatter pheromone randomly. On the other hand, when most agents exploit the same trail, old trails evaporate and the entropy decreases.

Example of fast convergence to a sub-optimal trail

A simulation showing commonly witnessed result is presented here. A directional bias with medium impact is used: a coefficient of $\eta = 0.3$. One can see in Figure 4-10, that after a brief initial exploration phase, some trails appear. Very early already, a strong foraging trail appears but also some smaller trials. Barely visible in Figure 4-10b, there is a trail between the nest on the left and the bottom right corner of the domain, being reinforced by a handful of agents. Eventually, all but one trail collapse. The occurrence of a trail collapsing can be identified by the local peaks in the entropy graph (Figure 4-11). According to that same graph, the best foraging trail is developed just over 300 seconds into the simulation. The very slight increase in entropy witnessed after 350 seconds is due to the foraging trail slightly displacing visible when comparing Figure 4-10c to Figure 4-10d. The displacement of the trail can be restricted when the agent is modeled with a shorter distance between the CR and deposition location.

Example with fast convergence to the global best trail

As a final demonstration, a very efficient simulation is presented. Here, a strong directional bias is used with a coefficient of $\eta = 0.6$. The resulting pheromone graphs are shown in Figure 4-12 that the unprofitable trails evaporate quickly after the experiment is started and the directional bias straightens out the curves in the main foraging trail. This analysis is confirmed by the evolution of the entropy of the pheromone graph as shown in Figure 4-13. One third into the simulation, the graph has completely converged to the minimal entropy.

4-7 Swarm configuration

In the previous section, an effective control algorithm was presented including a directional override function, steering the agents towards the nest. In practice, when the control algorithm is applied to real robots, such an override function may not be available. For example, when robots operate in 'stand-alone' mode, with real pheromone dispensers and without the central computer program, robots need special hardware, such as an Inertial Measurement Unit (IMU) to know its position relative to the nest. When robots do not have such hardware, control override cannot be used.

As a guideline for the hardware design of the swarm, a specific parameter sensitivity analysis is performed. Here, it is assumed that with a given, fixed control algorithm on the robots, the only design parameters are the pheromone evaporation rate and the number of robots to use. Two cases are tested. First the Weber control method without directional bias is tested. Second, the Weber controller with a large directional bias is tested. These two control methods are the best of their class as demonstrated in Section 4-6.

4-7-1 Results

The results of both control methods are displayed in Figure 4-14. The experiment duration is 600 seconds, all the experiment parameters are similar to scenario 2 in Table C-1 (see column 'scenario 3'), the control parameters are according to Table 4-7. One can see that, as expected, the control method with override outperforms the other over the whole range of variables.

For the control method without directional bias, the sweet spot for agent swarm size and pheromone half-life time seems to be between 80 and 120 agents at a half-life time of 60 to 120 seconds. A positive result is that the performance of the swarm is reasonable even for small swarm sizes, given that the pheromone does not evaporate rapidly. On the other end of the spectrum, when pheromone does not evaporate, the decay of unprofitable trails is delayed, impeding the performance of the simulation.

For the control method with direction bias, the results are very good over the whole range. Only when very few agents are used with a fast evaporating pheromone, do the results suffer. The deteriorating performance with very many agents with very slow evaporation may indicate that agents are trapped in cyclic trails that do not evaporate fast enough.

For comparison, a graph displaying the relation between the half-life time $t_{\frac{1}{2}}$ and evaporation rate ρ (based on decay per second) is shown here in Figure 4-15.

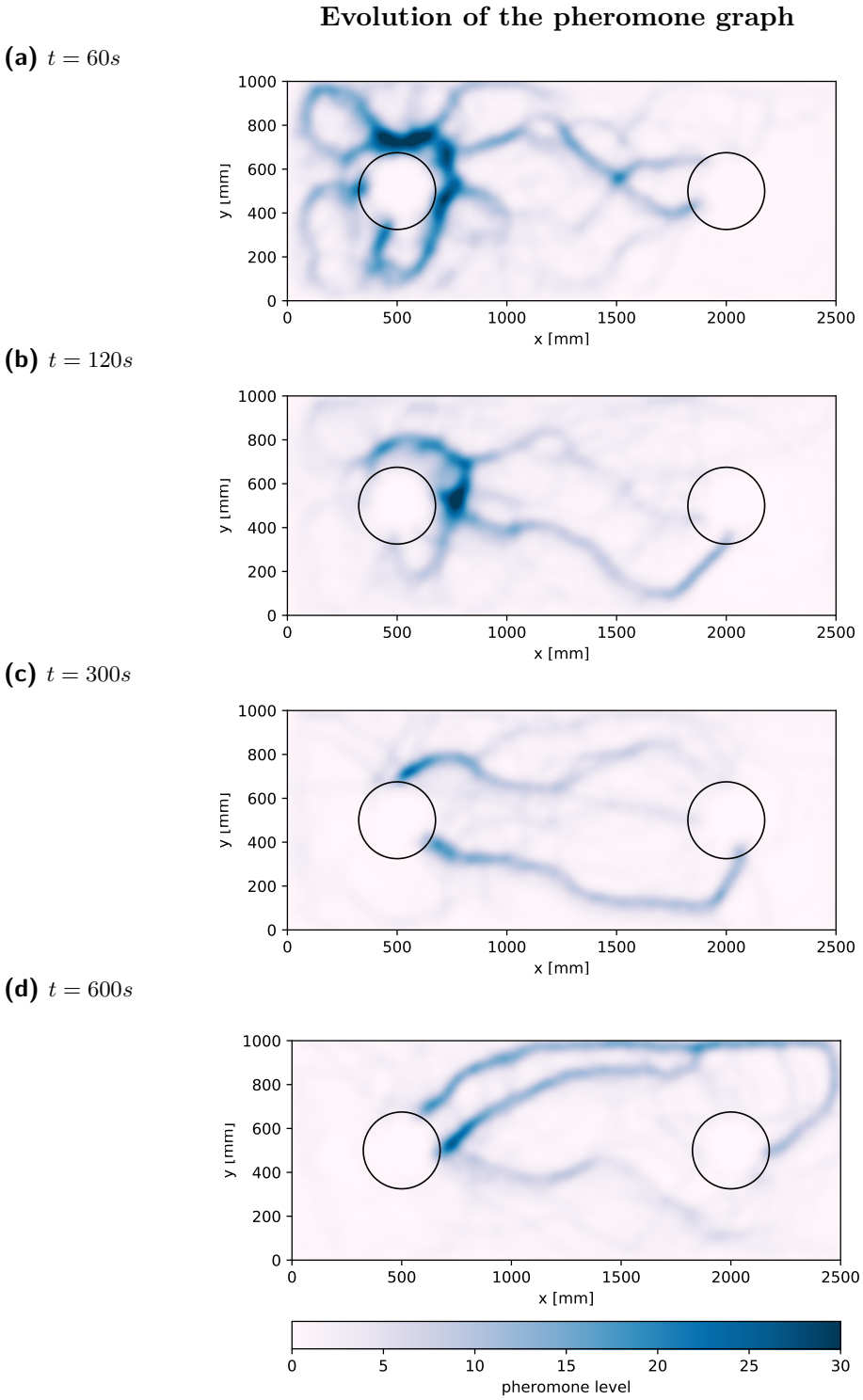


Figure 4-4: Evolution of the pheromone graph of the best benchmark simulation. The circle on the left represents the nest entrance, on the right the food source.

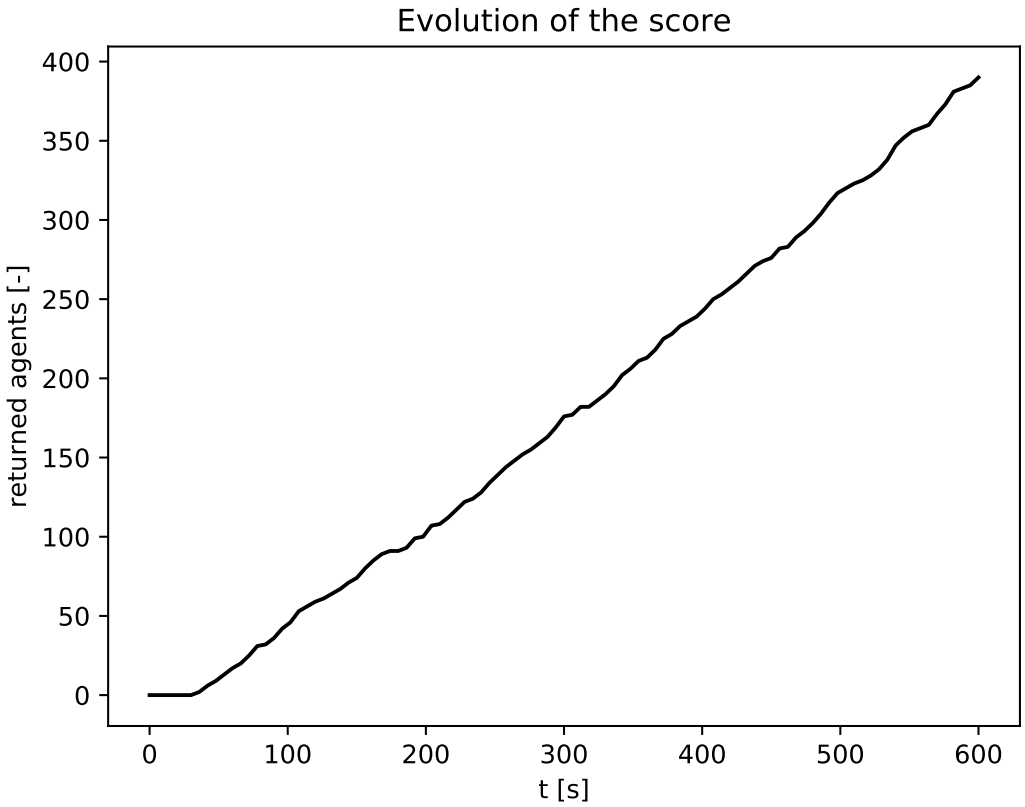


Figure 4-5: Number of agents making a successful round-trip between the nest entrance and the food source, corresponding with the pheromone graph presented in Figure 4-4.

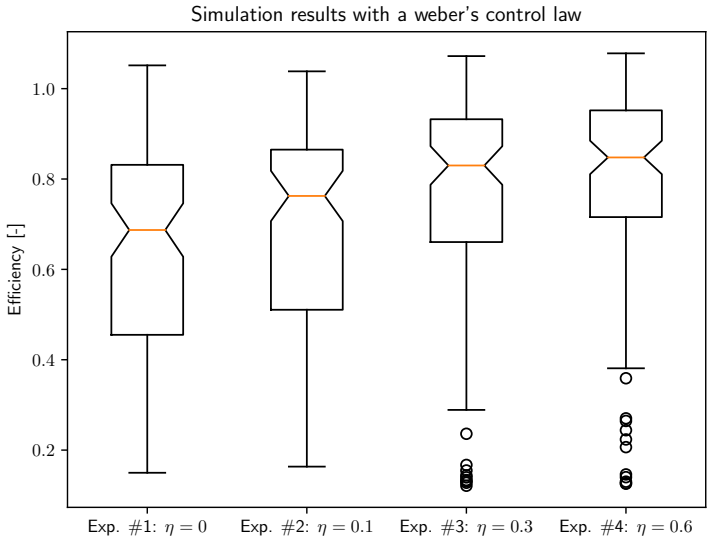


Figure 4-6: Efficiency rating for the different cases using the Weber's law inspired controller. The corresponding statistics are presented in Table 4-8.

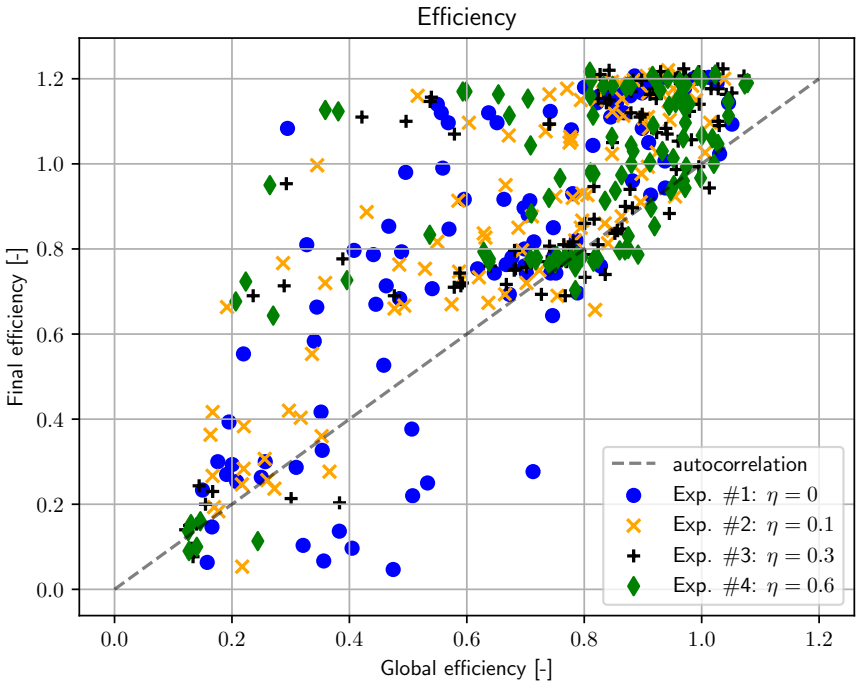


Figure 4-7: Scatter plot showing for different simulations and cases the achieved efficiency in the last stage of the experiment compared to the overall achieved efficiency of that simulation.

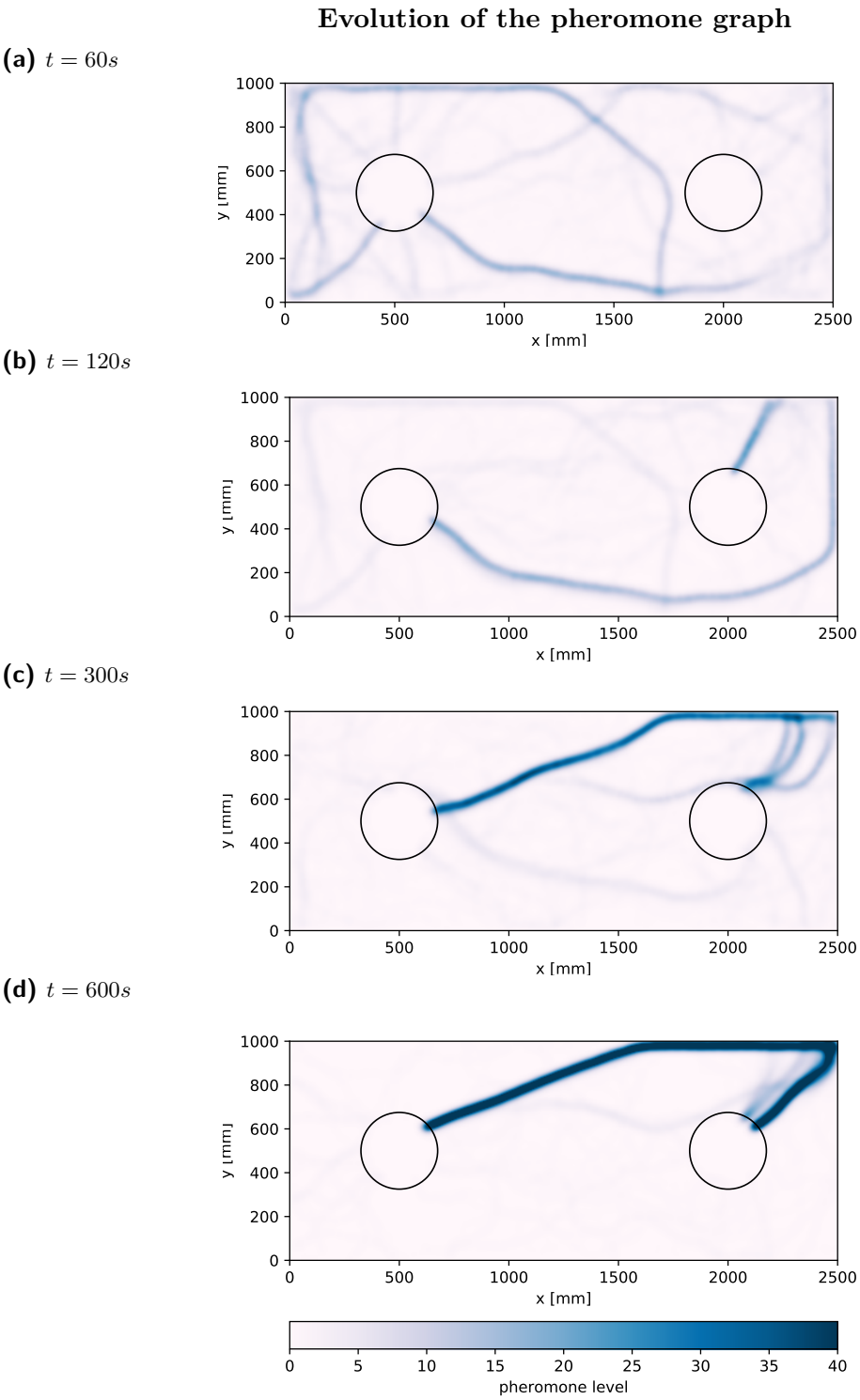


Figure 4-8: Evolution of the pheromone graph of an underperforming simulation using the Weber control model without override. The circle on the left represents the nest entrance, on the right the food source.

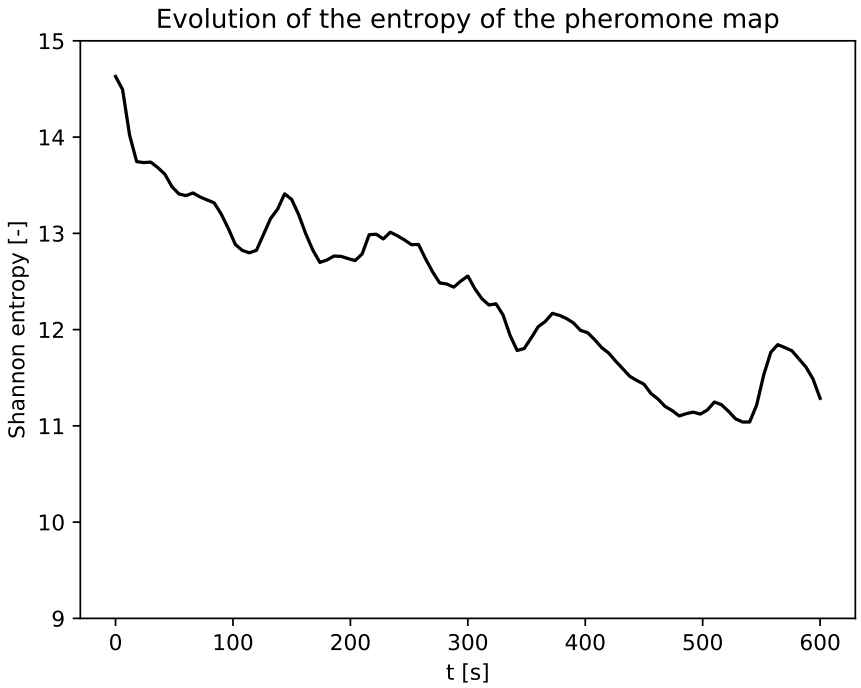


Figure 4-9: Evolution of the entropy of the pheromone graph corresponding to the simulation shown in Figure 4-8.

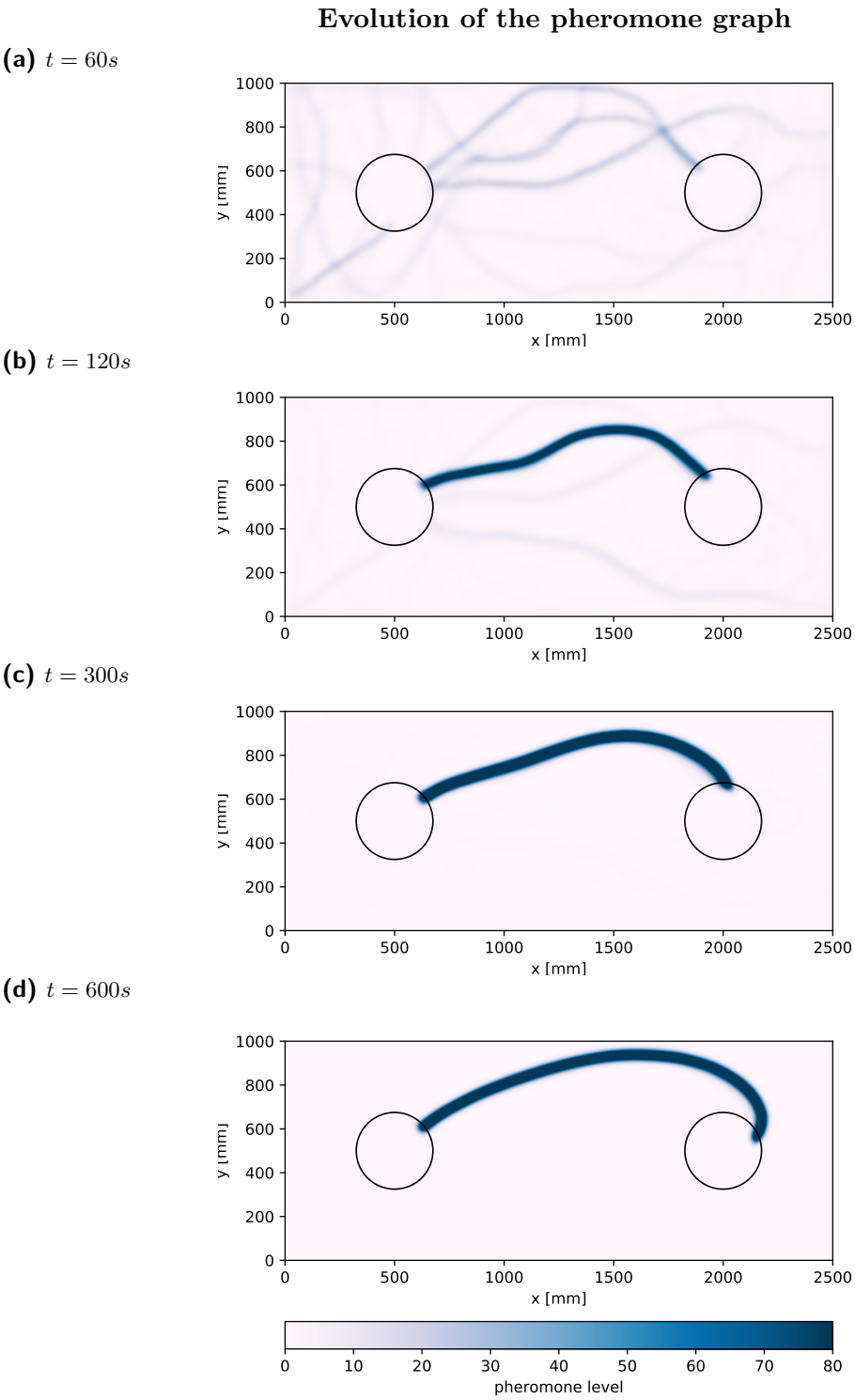


Figure 4-10: Evolution of the pheromone graph of a representative simulation using the Weber control model with a modest amount of override. The circle on the left represents the nest entrance, on the right the food source.

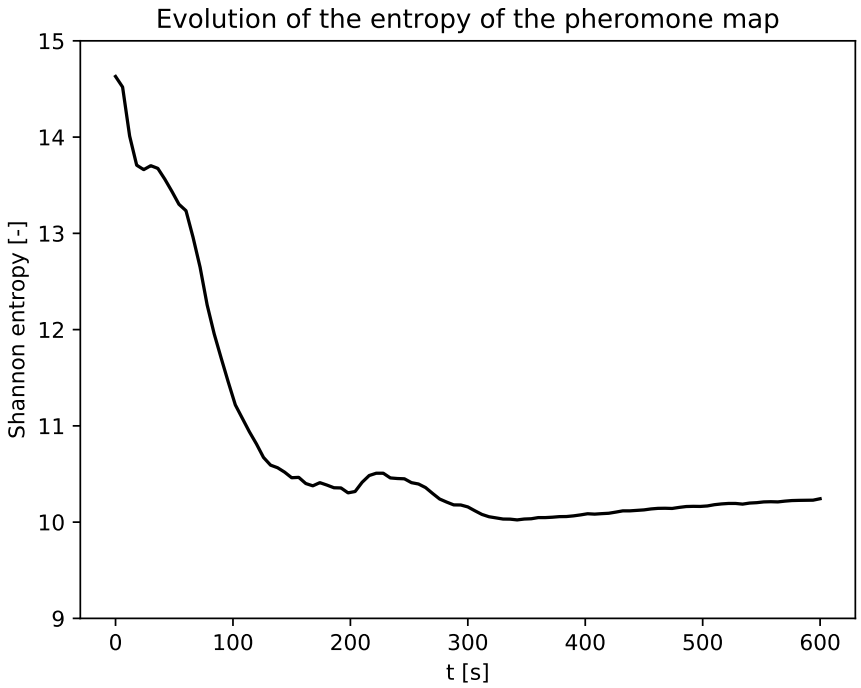


Figure 4-11: Evolution of the entropy of the pheromone graph corresponding to the simulation shown in Figure 4-10.

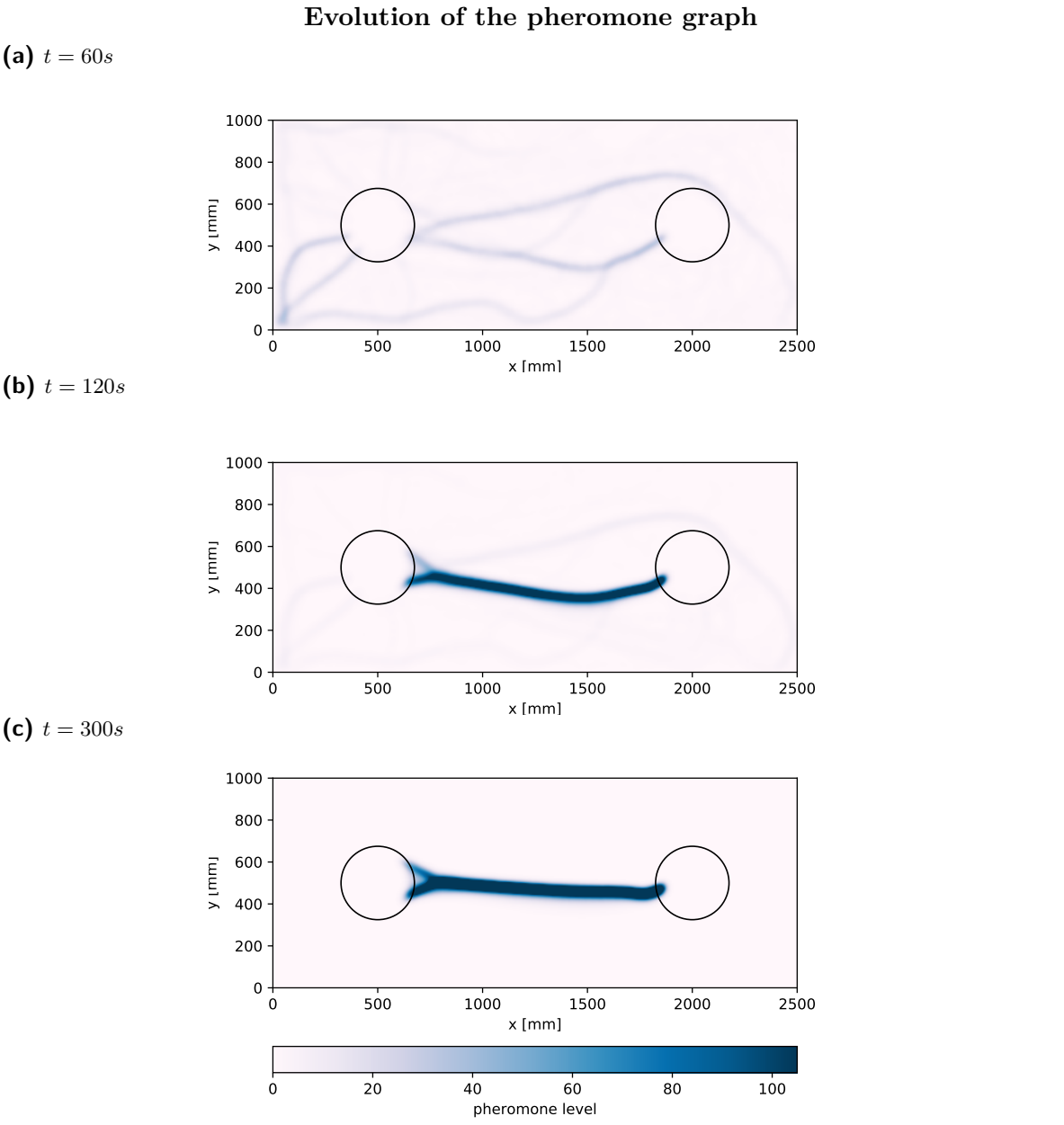


Figure 4-12: Evolution of the pheromone graph of a representative simulation using the Weber control model with a large amount of override. The circle on the left represents the nest entrance, on the right the food source.

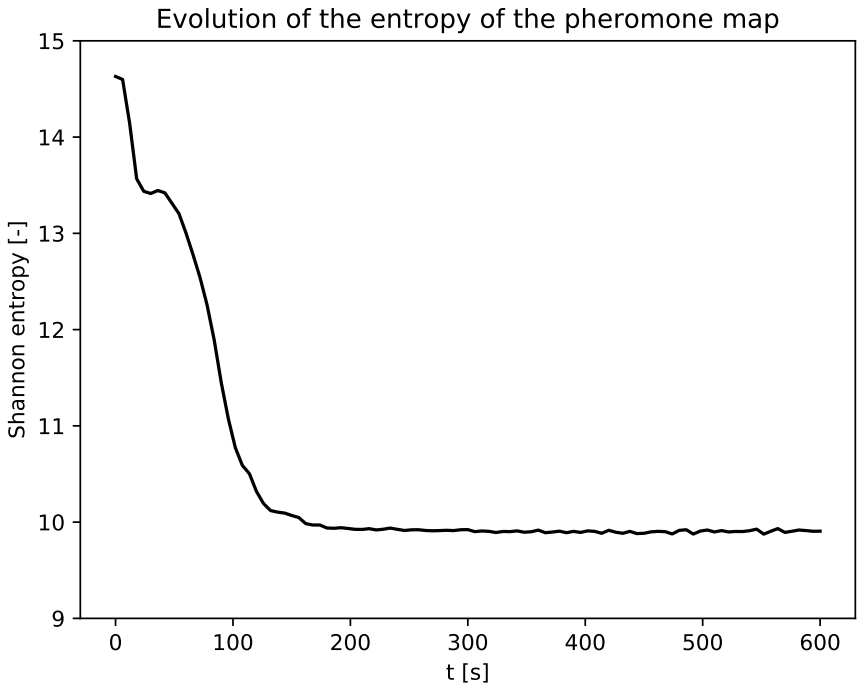
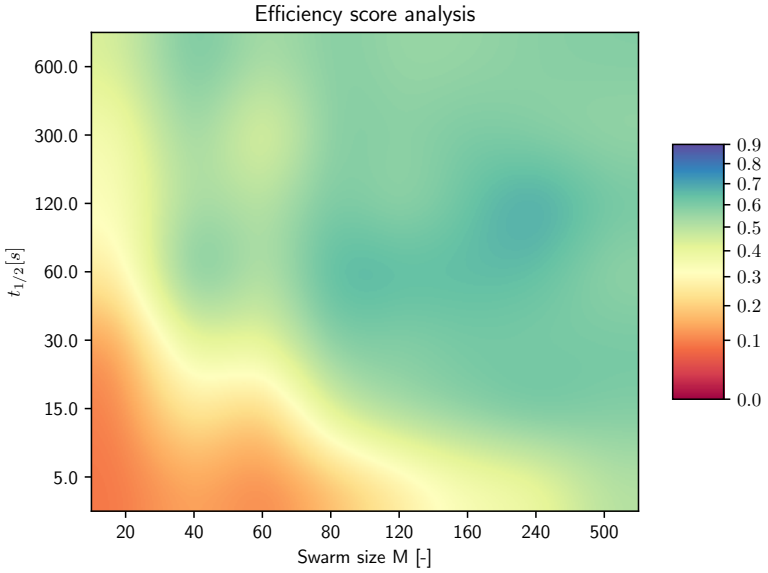
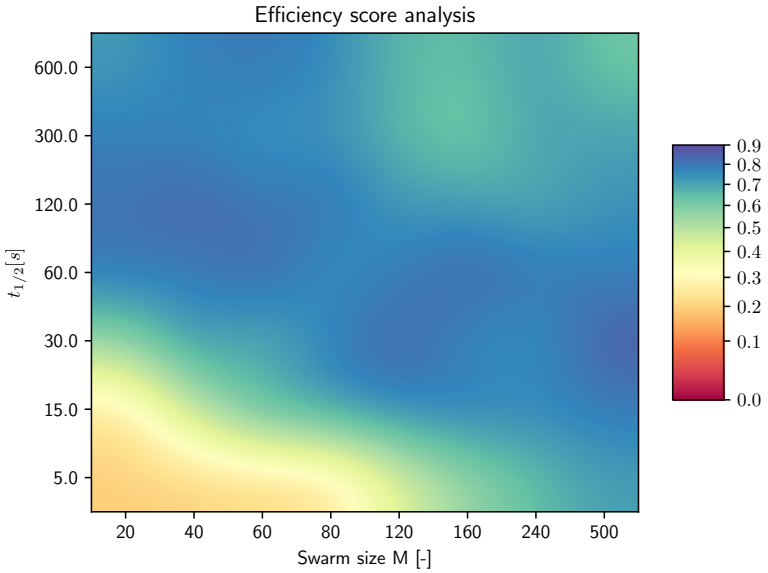


Figure 4-13: Evolution of the entropy of the pheromone graph corresponding to the simulation shown in Figure 4-12.



(a) Results for the 'Weber' control method without override.



(b) Results for the 'Weber' control method with override factor $\eta = 0.6$.

Figure 4-14: Visualization of the average efficiency for scenarios with different pheromone half-life times and swarm sizes. The experiment duration is 600 seconds. Results are based on the mean efficiency score obtained from 50 experiments per sample.

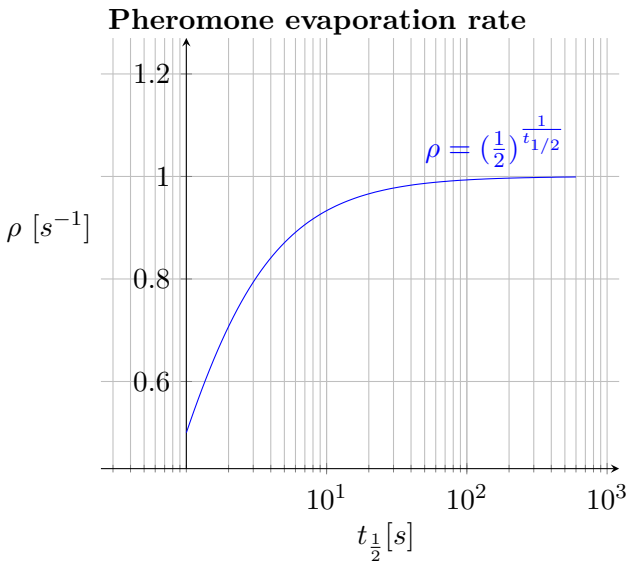


Figure 4-15: Pheromone evaporation rate as a function of its half-life time.

Chapter 5

Discussion

With all the recorded simulations, one can see that the control method from the second scenario (Weber) is a significant improvement over the simplest one. Most design parameters are treated as constant over the different tests. The influence of the control method and later also the colony size and pheromone evaporation rate was analyzed. In this chapter, a reflection on the results is presented and a set of possible applications for the control algorithm is suggested.

5-1 Simple vs Weber model

From the experiments in Chapter 4 it is apparent that the Weber control model outperforms the simple control model by a significant margin. Visually, the most prominent difference is that with the simple model, agents are unable to break away from the boundary, despite the randomness in the movement. Second, the agent response to pheromone changes with the number of depositions: the more agents in the simulation, the stronger the response of the agents to pheromone. That is the reason that agents are prone to being trapped in cyclic trails at the start of the experiment using the simple model. When many agents enter the domain at the nest entrance and start depositing pheromone, the concentration of pheromone around the nest is very high. In fact, the pheromone concentration is so high in some cases, that agents are directed to the nest again due to the attraction of the pheromone. They keep encircling the nest until the time dependent pheromone deposition prevents further reinforcement of these trails.

The very nature of the Weber control model prevents agents from being trapped around the nest, but again, there is a caveat. With the Weber control model, the agents are more sensitive to faint traces of pheromone trails. Hence, when agents leave the nest, they are susceptible to follow an other agent's trail. This behavior makes the exploitation less efficient and causes agents to be trapped in other unprofitable trails.

5-2 Entropy

The entropy as introduced by [4], proved a reliable indicator for the estimation of the convergence and strength of the foraging trails. In contrast to the discrete case, the decrease of the entropy during the simulation, is not as spectacular. Even when a very strong foraging trail has emerged (Figure 4-13), only a decrease of approx. 35% in the graph entropy is witnessed.

5-3 Time based depositing

The agent state based deposit function is introduced in Section 3-7-1. During the tuning of the control algorithms, it proved a vital instrument for improving the efficiency of the swarm. Especially for the simple control method, but to some extend also the Weber model, benefit from this decay in pheromone contribution from the agents. Without this feature, the unprofitable, cyclic trails that occur with most simulations at the start of the experiment, keep receiving reinforcing pheromones from the agent traversing these trails. By doing so, these trails grow stronger over time and manage to capture more agents while the trails themselves only loop in cycles.

5-3-1 Pheromone covariance

A difference on the hardware side when comparing the two control models, is that the simple control method performs better with a larger spread in the pheromone deposition (increase in deposition variance). With more widespread pheromone distributions, it is preferred to have more separation between both sensors, hence the differences in sensor offset and deposition covariance (Table C-1).

5-4 Override

The second instrument in optimizing the efficiency of the swarm, is the directional bias ascribed to agents who successfully found the food as introduced in Section 3-7-3. This instrument boosts the performance of the Weber control model but is less effective in the simple control model. A reason may be the fact that with the simple control model, the majority of the agents is stuck at the boundary of the domain. In order for the directional memory to work, an agent has to find the food in the first place.

With the Weber control model, the override is a lot more efficient as discovered in Section 4-6. This result can be used in designing an algorithm for real robots. When the hardware allows for relative position determination, the directional bias can help the establishment of a pheromone trail. On the other hand, these tests are conducted in an unobstructed rectangular domain. In the case of a domain with obstacles, the directional override may prove counterproductive. In such a case, reverting back to biomimicry may prove useful as both certain species of ants and bees are suspected to use landmarks for navigation in obstructed domains [38–40].

5-5 Agent layout

As briefly mentioned in Section 4-6-3, when the actuator is placed at some distance of the CR, it can happen that slightly curved foraging trails drift away over time and can even collapse despite strong reinforcements. This is caused by the agent rotation. When it is in the center of the pheromone trail and rotates about its CR, the actuator swings slightly to the side of the trail, opposite to the agent turning direction. This causes the accumulation of pheromone on the outer part of a curve in the foraging trail, thereby displacing the trail over time. Designing the agent such that the actuator is placed very close to the CR, and a suitable high steering gain (or smaller time steps) such that agents do not 'understeer' in the corners, prevent the trails from displacing over time.

5-6 Parameters

The scenarios that were tested in Chapter 4, all had most properties regarding the domain and the agents in common. The agent settings and control methods are tuned to perform optimally in the given domain. Some considerations about the influence of the domain size on the required swarm characteristics are discussed here.

5-6-1 Exploration

The noise parameters dictate how fast agents explore a domain, and consequently, how fast they can discover the nest. If the domain would be larger, the characteristics of the noise are likely to be adjusted. During the testing of the control methods, it became obvious that the higher the covariance of the noise, the slower the exploitation. The reason behind this statement, is that more noise, causes the agents to make many random turns. The agent density (agents per m^2) remains relatively high in the neighborhood of the nest, a desirable property when the food is located close to the nest. Less noise allows agents to spread out more quickly. The downside is that with less noise, agents are more prone to follow each other's trail. This allows agents to explore more distant locations but they do so less thoroughly.

5-6-2 Trail formation

In order to have the swarm develop a strong foraging trail, enough pheromones are to accumulate such that it attracts agents in the vicinity. When the domain is larger, more pheromone is to be deposited to keep the pheromone density in the graph relatively constant. With the evaporation, it may be more difficult to establish foraging trails over longer distances.

In order to allow foraging trails to establish even in larger domains, more pheromone is to be added during the simulation. There are, globally speaking, three sources that

dictate the amount of pheromones on the graph. First, there is the evaporation rate. It is likely that with a larger domain, the pheromone evaporation rate can be a bit slower. Second, placing more agents in the simulation will naturally increase the amount of pheromone additions. An additional benefit is that more agents will speed up the exploration phase as well. Third, there is the deposition quantity of the agents. When the domain is larger, the time it takes to travel between the food and the nest entrance increases. Hence, the decay rate of pheromone deposition as described in Section 3-7-1, should be adjusted accordingly.

5-6-3 Event triggering conditions

With the simulation the discrete of Jarne Ornia [4], in one situation agents deposit pheromone according to an event triggered condition. The entropy of the graph is estimated by the entropy of the links adjacent to the agent location. If the local entropy is higher than a threshold value, the probability of an agent depositing pheromone on the link it traverses is less than one. The advantage is that an agent dispenses less pheromone during the experiment. With real robots, this may translate in a longer experiment duration if the pheromone supply is the limiting factor.

In the simulation presented in Chapter 4, the agents deposit a quantity of pheromone dependent on the state of the agent. The longer an agent is without successfully reaching the nest or food, the less pheromone it dispenses. Although this can be regarded as a form of event triggered pheromone deposition, the act of depositing pheromone does not depend on the local perception of pheromone on the graph.

5-7 Applications

Compared to robots with real marking chemicals, keeping a virtual pheromone map (graph) implies an extra source of computational complexity. Yet, there are advantages to this approach. For real robots in a room with object tracking equipment, the robot position can be presented to a central computer program that simulates the deposition of pheromone. Instead of having the robots sense real pheromone, a robot queries the concentration from a coordinating computer program. This approach increases the software complexity with the benefit of a reduction in hardware complexity: robots do not need to carry a pheromone dispenser and relevant sensors.

This methodology can have an application in the real world as well. Consider a factory where autonomous navigating robots operate in the same domain as humans. Such is the case at Prodrive Technologies [41] where robots navigate the office spaces to transport supplies to manufacturing stations. The path planning of these robots is a continuous challenge as furniture, bags and humans can block existing routes. A stigmergy-based approach with a virtual pheromone map could be a perfect solution.

Besides from the theoretical point of view, a swarm of ant inspired robots can prove useful. One could put a swarm of real robots depositing real pheromones (such as

paint, water, alcohol or heat or any other marking agent) to work. In an unknown environment, where it is easier or more safe to release a swarm of robots instead of human exploration, these artificial ants can explore the domain. Such an application could be assisting in search and rescue missions in cave systems or assisting in the path planning for a larger, more expensive and vulnerable robot such as the mars rover [14].

Chapter 6

Conclusion

In this thesis, a decentralized control method for a swarm of robots is presented. The methodology is based on earlier work of foremost Perna et. al. [10], Alers and Tuyls et. al. [12] and the MSc thesis of Jarne Ornia [4]. By applying the concepts of pheromone based navigation (Perna) to the exploitation in a graph (Jarne Ornia), a suitable control method is presented such that it can be applied to a swarm of real robots (Alers).

It is demonstrated that a decentralized algorithm based on pheromone perception alone, is sufficient for coordinating a robot swarm to the task of establishing a trail between two locations. A necessary condition is that the agents deposit less pheromone over time when they do not successfully make a trip between the nest entrance and the food. When the agents are ascribed a sense of directionality, the performance of the swarm can be improved further.

As an additional benefit, the control algorithm could be implemented on individual robots. It has also been suggested that the centralized control method, with a central program that keeps track of pheromone distributions, can be applied to solve real world navigation problems as well.

The control methods are developed and tested in a computer simulation program that is presented together with this thesis. The computer program is publicly available for use and interested people are invited to further develop it. The performance of the program in terms of processing power demands, was sufficient to allow a big number of simulations be executed in a short amount of time. This simulator allows for a number of future research directions on the application of ant based navigation.

Chapter 7

Future work

In the preceding chapters, a suitable set of control laws is presented and analyzed, that can be applied to real robots in a decentralized way. Unfortunately, the realization of the implementation proved too time consuming to include in this work. Besides from that, the main question was whether we can design and simulate a control algorithm based on local available information alone, and if this is sufficient for swarm robot control. It is shown that the answer is: yes we can. But as is often the case, answering one question raises many more.

7-1 Real robots

Maybe the most prominent question, is whether the control method works on real robots. In order to demonstrate a proof of concept, an experiment with a swarm of robots is to be carried out. This may be executed in three phases. First, the correlation in behavior between virtual agents and real robots is to be assessed. This assessment can be carried out by replacing the agents in the simulations with robots. The positions of the robot is not calculated but obtained from a motion capture system. The simulation software is to send an instruction for rotation and translation to the robot, and after some time, obtain the new robot position. Such an experiment should yield valuable information about the correlation between the anticipated robot's next position and the actual robot position. It may be possible that the simulated sensor noise can be replaced by the randomness following from the mismatch between the anticipated and actual robot position.

The second stage of the process of testing the algorithm on real robots, is by letting the robots actually calculate their own instructions. Then the program doing the central coordination only keeps track of the pheromone map. Through a communication protocol, it sends all the robots the pheromone concentration at their positions.

The final stage would be to perform an experiment with the control algorithm with actual pheromone, such as Alers, Tuyls et. al. [12] have demonstrated, but at a much larger scale.

Collision avoidance

As in the simulator, agents are allowed to physically occupy the same space, and do so very frequently during exploitation. It is expected that real robots will often collide, possibly impeding the simulator performance. The works of Couzin and Franks [26], who have demonstrated a model where contact among agents, and agglomeration of pheromone acts as a direction repellent (negative feedback). By doing so, agents are witnessed to form lanes, separating the traffic flow by direction. These methods may prove useful when applied in conjunction with the control method presented in this work.

7-2 Event trigger

As briefly mentioned in Section 5-6-3, an event triggered method for pheromone deposition may be useful especially when real robots with a limited supply of pheromone are used. Agents with two sensors cannot reliably estimate the convergence of the graph by estimating the entropy, but they can do so based on the sensed pheromone levels. One can deduct from the graphs shown in Section 4-6 that the pheromone levels on established trails are in the order of 10 to 50 times higher than the pheromone levels on the map that are not part of the trail. A cue to reduce the pheromone addition such as in [4] could be when the mean of the perceived pheromone levels is higher than some threshold value.

7-3 Domain

The simulator is only capable of simulation rectangular, unobstructed domains. In such an environment, the override functionality proved very successful in accelerating the swarm performance. During the testing phase of the software development, it was noticed that there may be a relation between pheromone decay rate, agent speed, deposition decay rate and domain size. In short, the longer it takes an agent to travel between the food and the nest (based on domain size and agent speed), the slower the decay rates of the pheromone deposition and deposited pheromone aught to be. This suspicion should be further investigated if the control methods are to be applied to domains of arbitrary size.

Further, the override functionality may be less suitable in domains with obstacles. A possible expansion for the simulator software may be to include domains with obstacles.

7-4 Dynamic control algorithm

The current control modes, and those of all who have modeled artificial ants, use a uniform swarm configuration. Yet, in nature, members of the colony can have different functions, translating in different control characteristics [6, 39, 40, 42, 43]. One could leverage this principle by designing the control algorithm non-uniform. Model some agents for efficient exploration and have these enter the domain before the exploitation agents join the simulation. As Reid et. al. [18] suggest, algorithms based on biomimicry, can benefit from stronger biomimicry.

Taking this one step further, if the simulations are computed fast enough, it may be feasible to generate a control strategy using a reinforcement learning approach with a cost function based on the computed efficiency of the simulation. This would automate the tuning of a general controller tuned to function in a variety of domain layouts, such as when obstacles in the domain are considered.

7-5 Simulator improvement

As a final note on the simulations, all experiments were performed with a time step of $\delta_t = 0.3s$ as this was the maximum allowable step given the agent speed. As the agent orientation is based on the rotation speed, which is influenced by noise, smaller time steps would result in less covariance in the agent heading considered as a time series. In other words, the smaller the time steps, the less influential the noise. In order to generalize the control strategy for different time steps, the noise has to be scaled accordingly.

Appendix A

theANT3000 - GUI

The manual of `theANT3000` is found on Github [44]. Presented here is a brief overview of the workings of the simulator and the relation between the GUI input and the simulation settings and control algorithms in Chapters 2 and 4.

Program working

The GUI of `theANT3000` is a cosmetic layer between a C and Python3 base program and the user. Through `theANT3000`, simulations according to the rules presented in this work can be performed. Each time a simulation is started, a unique identifier number, the `ID`, is presented in the main window. This `ID` can be used to retrieve simulation results from the database.

For each simulation, all the related settings stored in an SQLite database and can be retrieved through the `ID`. The state of the graph is explicitly not stored in a database as this would occupy too much diskspace. Instead, the agent position, orientation and pheromone deposition quantity is stored to a table in the database. With this information, the pheromone graph can be reconstructed to the original state at any step in the simulation.

Next to the agent location, the evolution of the number of returning ants to the nest and the evolution of the entropy of the pheromone graph is stored for analysis and comparison among different simulations.

Main settings

A screenshot of the GUI is shown in Figure A-1. In the top-left corner, there is the Menu with one button, Quit(). If the program is terminated through the action Quit(), the last-used settings are stored and displayed at the next time the program is launched.

The highlighted field #1 (Sim settings) shows 3 buttons and a text input field in the top-left corner. The input field allows one to enter the `ID` (integer) of a simulation. By default, the simulation with the highest efficiency score is shown when available.

The **Replay** button starts the visualization of a previous recorded simulation, specified by the `ID`. If all the simulation steps have been recorded, the visualization is animated, else only the statistics about the number of returning ants to the nest (score) and the evolution of the entropy is shown.

The **Copy** button allows one to copy the settings of the simulation specified by the `ID` to the corresponding input fields.

The **Run** button, starts a new simulation. The efficiency score and its `ID` is shown in the main textbox. If **Recording** is checked, all steps of all agents are recorded to the database so that the simulation can be replayed at a later time. If the checkbox **visualize** is checked, an animation of the simulation will start right after the program is finished computing it.

DB write interval specifies the number of simulation steps to be stored in the RAM before it is written to the database (only applicable when **Recording** is checked).

The remainder of the input fields are elaborated on in the tables below, some input fields define multiple parameters.

Table A-1: Parameters as displayed in Figure A-1.

ID	Name	Par.	Reference	Comment
1	Colony size	M	Definition 3.2	
	Time step	δ_t	Definition 3.1	
	Steps	N	Definition 3.1	
2	Antenna length	l	Figure 3-1	
	Actuator distance	d	Figure 3-1	
	Sensor offset	Γ	Figure 3-1	
	Ant speed	v	Equation (3-3)	
3	Sensor activation	$g(w_{i,j})$	Section 3-6-1	Linear(1:1) or ReLu.
	Breakpoint	z_{br}	Definition 3.19	Only when using ReLu perception.
	Control function	$f(\vec{\tau}(k))$	Sections 3-6-1 and 3-6-2	
	Gain	α	Sections 3-6-1 and 3-6-2	Feedforward Steering gain.
	Noise type	$\epsilon_{1,2}, \varsigma$	Equations (3-23) and (3-29)	Gaussian or Telegraphic noise series.
	k (weber reg.)	κ	Definition 3.18	Regularization parameter when using weber control function.
	Noise gain/cov #1	σ_1^2, α_1	Definitions 3.18 and 3.22	Covariance in the case of white noise, feedforward noise gain for telegraphic.
	Noise cov 2	σ_2^2	Definition 3.18	Only for weber model.
	Noise parameter	β	Equation (3-29)	$T_s \sim \text{Exp}(\frac{1}{\beta})$
4	Control override	$g_m(k)$	Equation (3-34)	Enable boolean.
	Override factor	η	Figure 3-4	
	Override time	$t_{\max}$	Figure 3-4	

1
Slim settings

2
Agent properties

3
Control settings

4
Override

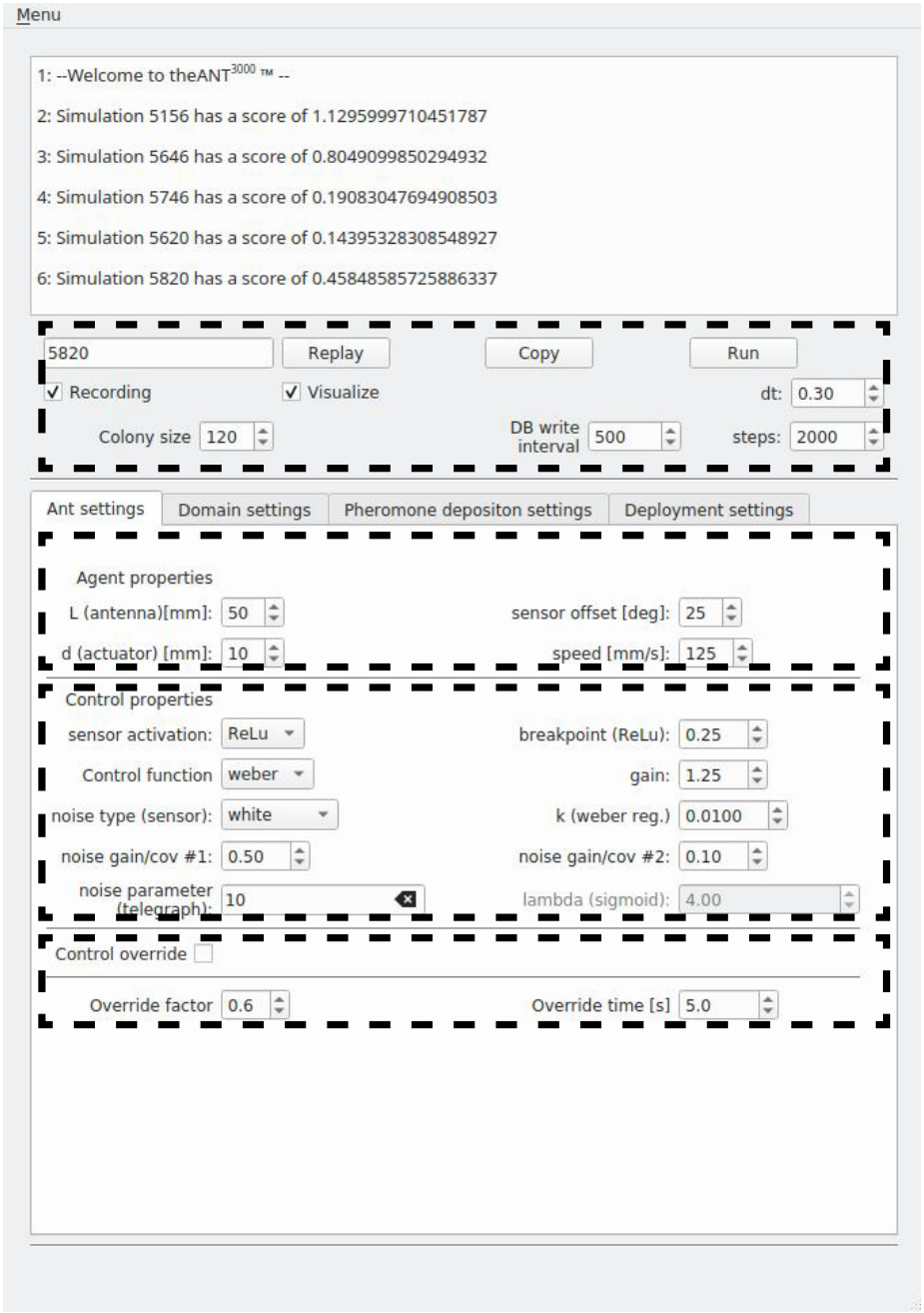


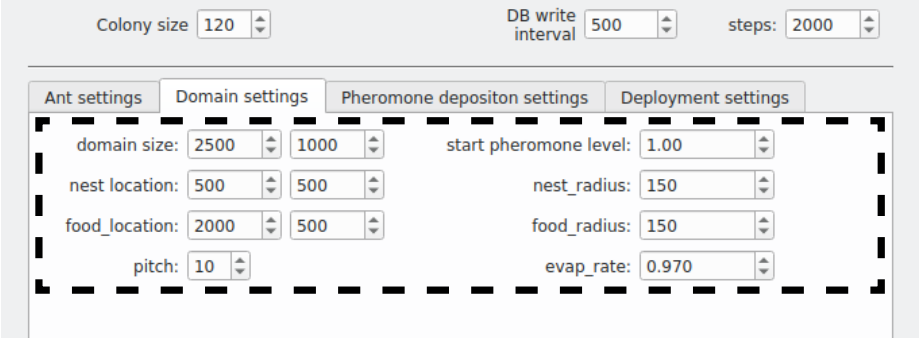
Figure A-1: `theANT3000` main view with ant settings.

Table A-2: Parameters as displayed in Figure A-2.

ID	Name	Par.	Reference	Comment
5	Domain size	X, Y	Table 3-2	
	Start pheromone level	$w_{i,j}(k=0)$	Section 4-3	
	Nest location	x_n, y_n	Table 3-2	
	Nest radius	R	Table 3-2	
	Food location	x_f, y_f	Table 3-2	
	Food radius	-	Table 3-2	Same as nest radius.
	Pitch	Δh	Table 3-2	
	evap_rate	ρ	Proposition 3.1	
6	Deposit covariance	σ^2	Proposition 3.2	
	Deposit quantity	q_0	Equation (3-28)	
	Return factor		Section 4-4	State based multiplier for q_0.
	Deposit function	$q_m(k)$	Equation (3-28)	
	Gaussian digits	ι	Proposition 3.2	
	Decay parameter	γ	Equation (3-28)	(name confusion ...).
7	Deployment		Section 4-2-1	Just 1 option is defined.
	Timing		Section 4-2-1	Gamma, uniform or instant.
	k (timing Gamma)		Section 4-2-1 [35]	k in $Gamma(k, \theta)$.
	teta (timing gamma)		Section 4-2-1 [35]	θ in $Gamma(k, \theta)$.
	t_max (timing uniform)		Section 4-2-1	a in $\mathcal{U}(0, a)$.

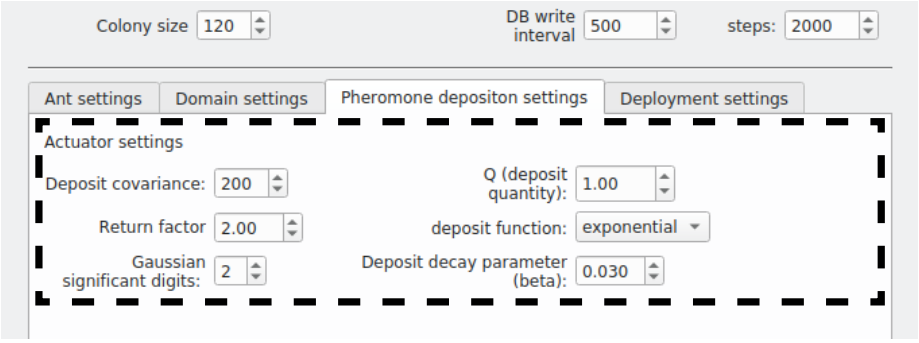
(a) Tab: Domain settings.

5
Domain shape



(b) Tab: Pheromone deposition settings.

6
Pheromone



(c) Tab: Deployment settings.

7
Deployment

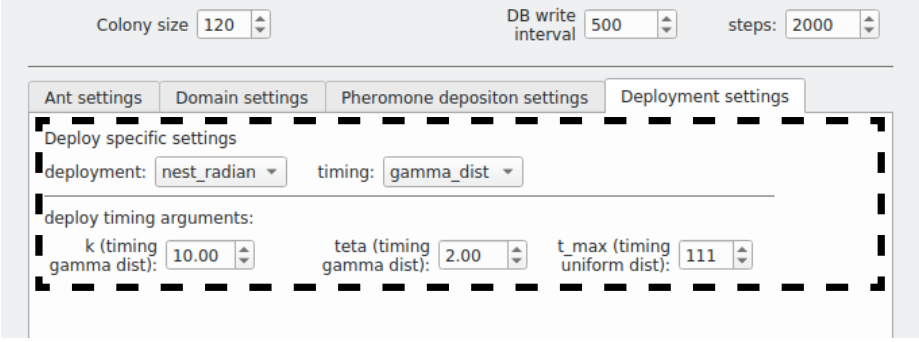


Figure A-2: `theANT3000` secondary views.

Appendix B

theANT3000 - Performance

The simulation software, called `theANT3000`, is publicly available on Github [44]. During the development of the software, it became obvious that for a function point of view, the computation time for the simulation has to be minimal. The stochastic nature of the simulation requires large amount of trials so that the statistics of many experiments can be analyzed.

B-1 Simulator performance

Although computers are becoming increasingly faster, large multi-agent simulations are still challenging to compute efficiently. For example, Perna et. al. [10] mention that it takes a computer 5 minutes to compute one second worth of simulation data. During the development of `theANT3000`, special attention was paid to the optimal use of computational resources. The first version of `theANT3000` was completely programmed in Python3, using the Numpy API for performing most mathematical operations. Compared to early versions of the software, the current version is roughly 50 to 80 times faster.

B-1-1 Cython

The first heap of the performance gains was achieved by compiling the python code. Traditionally, Python is an interpreted language [45] and such a script is interpreted and converted to cpu instructions on run-time. By compiling the python code, it is transformed into machine language at compilation time which omits the need to interpret the script on run-time. An execution time speedup of 1.5 to 5 times is expected in this step.

C types

A second benefit of using Cython, is the possibility to convert generate C and C++ code. All performance critical components of the simulation are written in Cython using static type definitions and subsequently converted into C and C++ code. Consequently, the resulting C and C++ files are compiled into machine code. This conversion accounts for approximately half of the total performance gain. Especially the loops with nested logic (if-else if-else) profit from static typecasting.

Pointers

Most of the computations involve the state of the agent. By defining a C++ vector containing c-structs with all state variables it is possible to pass only the memory location of the agent state to a function. This way, a function does not have to allocate memory for temporary variables each time the function is called.

Compiler instructions

When the generated C and C++ files are compiled, a specific set of compiler instructions is used that speed up the execution time of the program considerably:

- **Boundscheck = False** - This instruction makes sure that when an array memory element is accessed, the program does not verify whether the accessed location does belong to the array.[1]
- **Initializedcheck = False** - When a variable is accessed, the program does not verify that the variable has been initialized already.
- **Nonecheck = False** - The program does not verify if an accessed array element is not 'none' (check validity of the corresponding memory block).
- **Wraparound = False** - Prevents the use of relative array indexing: e.g. accessing element A[-1] is not possible.
- **cdivision = True** - The program is compiled using the C machine instructions to calculate a division instead of the default Python method as the latter includes a bunch of performance deteriorating safety checks.

Hard-coded loops

Ultimately, hard coded functions for manipulating large matrices are used. Because of the compiler instructions and the possibility to use all logical cpu cores, the code is considerably faster. For example, the sum of a matrix, computed by using multiple parallel loops to read all matrix elements and omitting any precautionary sanity check, is much faster than using the Numpy `sum()` of a matrix.

[1] Accidentally attempting to access element $N + 1$ of an array of length N leads to hilarious results as the program happily accepts anything read at the supposed memory location of element $N + 1$.

B-1-2 Runtime analysis

In order to gain an understanding of where the bottlenecks are, a runtime analysis is performed. The result is shown in Listing B.1 [2]. Here, a single simulation of 2000 steps with 120 agents is performed on a domain with a grid spacing of 2mm and a pheromone covariance of 200.

Although timing the simulation influences the result (a timed simulation takes more than twice as long to compute compared to a non-timed simulation), the results are decisive enough to draw some conclusions. The interesting information is on the first lines: by far the most computation time (2.5 out of 9.5) is spent adding the pheromone contributions to the graph. A distant second, is the time spent pushing the results to the database. The remainder of the computation time (roughly 6 seconds) is mostly caused by the top 34 functions (line 9 to 43 in Listing B.1).

The pheromone addition function is called every time an agent adds pheromone to the graph. It scales with the number of agents, the number of steps and the number of graph elements are to be updated.

The number two, is the function used to push the result to the database. The scaling of this function is difficult to assess as it depends on the database speed. As every agent position, orientation and deposit quantity is logged, it is safe to assume this function run-time depends on the number of agents in the simulation and the number of steps.

The functions that are called 101 or 103 times, roughly worth a second of computation time, are those used to log the process of the simulation. Specific, the entropy evolution plot and the upper bound for the pheromone visualization. No matter how many steps in the simulation, these functions are always executed exactly 101 or 103 times. These functions scale with the size of the graph (linear, per axis) in both directions.

All other functions, executed over 200,000 times, are those that are executed at every step for every agent at least once. The run-time here, depends on the number of agents and the number of steps.

From this result, we can conclude that the dominant factor in the computation time is the number of pheromone additions to the graph. these additions scale linear with the amount of agents and number of steps, but quadratic with the pheromone covariance and inverse of grid spacing: by refining the grid to half the spacing, 4 times as much locations on the graph are updated by each ant addition:

$$\text{computing time} \sim \mathcal{O}\left(N, M, \left(\frac{1}{\Delta h}\right)^2, \left(\sigma^2\right)^2\right)$$

Furthermore, the results imply that the domain size does not contribute much to the computation time. Hence, somewhat larger domains can be simulated without noteworthy performance penalty. However, if the domain size doubles in both directions,

[2] Some file paths are removed in order to compact the output

while the grid spacing remains unchanged, the contribution of the pheromone graph functions ‘entropy‘, ‘sum‘ and ‘max‘ grow 4-fold and suddenly become noticeable.

Regarding the possibility of further optimization, the pheromone addition, entropy, and sum of the graph are all executed in C, using parallel computation. These functions performance scales very well in parallel execution on multiple cores. The timing was performed on a laptop running an Intel Core i5-8250@1.6GHz with 4 physical and 8 logical cores. Using a multi-core processor at a higher clock will yield better results.

Listing B.1: truncated cProfile runtime analysis of theANT3000

```
Sim 11799 has an efficiency score of 0.9775106051885485
11188573 function calls in 9.640 seconds

Ordered by: internal time

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
234555    2.460    0.000    2.662    0.000 cythonic/core/domain.pyx:114(add_pheromone)
4     1.102    0.275    1.102    0.275 cythonic/plugins/db_controller.pyx:40(executemany)
2001    0.892    0.000    0.892    0.000 cythonic/core/domain.pyx:74(evaporate)
2001    0.576    0.000    7.451    0.004 cythonic/core/sim_controller.pyx:106(sim_step)
103    0.493    0.005    0.678    0.007 cythonic/core/map.pyx:65(entropy)
469110    0.403    0.000    0.550    0.000 cythonic/core/domain.pyx:59(probe_pheromone)
234555    0.329    0.000    1.556    0.000 cythonic/core/ant.pyx:91(gradient_step)
971176    0.315    0.000    0.511    0.000 cythonic/plugins/functions.pyx:22(transform)
234555    0.299    0.000    0.443    0.000 cythonic/core/sim_controller.pyx:173(check_target)
234555    0.205    0.000    0.323    0.000 cythonic/core/ant.pyx:82(rotate)
2091729    0.199    0.000    0.199    0.000 cythonic/core/map.pyx:34(to_grid)
971176    0.196    0.000    0.196    0.000 cythonic/plugins/functions.pyx:11(rad)
243494    0.192    0.000    0.561    0.000 cythonic/core/ant.pyx:107(set_sensors)
234555    0.190    0.000    0.273    0.000 cythonic/core/sim_controller.pyx:160(check_attractiveness
    )
103    0.173    0.002    0.173    0.002 cythonic/core/map.pyx:41(sum)
101    0.156    0.002    0.156    0.002 cythonic/core/map.pyx:52(max)
234555    0.149    0.000    0.700    0.000 cythonic/core/domain.pyx:15(fill_observations)
1168997    0.142    0.000    0.142    0.000 cythonic/core/domain.pyx:46(check_pos)
2001    0.113    0.000    0.113    0.000 cythonic/core/sim_recorder.pyx:60(extract_antstate)
932321    0.106    0.000    0.106    0.000 cythonic/core/domain.pyx:27(check_bounds)
240694    0.095    0.000    0.237    0.000 cythonic/core/ant.pyx:100(step)
234555    0.095    0.000    1.651    0.000 cythonic/core/queen.pyx:46(step)
234555    0.095    0.000    0.126    0.000 cythonic/core/queen.pyx:94(assign_state)
234555    0.093    0.000    0.131    0.000 cythonic/core/ant.pyx:68(observe)
234555    0.088    0.000    0.127    0.000 cythonic/core/ant.pyx:56(calc_quantity)
234555    0.072    0.000    0.092    0.000 cythonic/core/map.pyx:14(span)
234555    0.049    0.000    0.049    0.000 cythonic/plugins/rotate_functions.pyx:30(weber)
240    0.047    0.000    0.047    0.000 cythonic/core/queen.pyx:102(noise_vec)
12    0.044    0.004    0.044    0.004 cythonic/plugins/db_controller.pyx:50(execute)
231755    0.039    0.000    0.039    0.000 cythonic/plugins/dep_functions.pyx:7(dep_exdecay)
231755    0.038    0.000    0.038    0.000 cythonic/plugins/sens_functions.pyx:15(observe_relu)
234555    0.037    0.000    0.037    0.000 cythonic/core/ant.pyx:113(increase_azimuth)
102163    0.031    0.000    0.031    0.000 cythonic/plugins/rotate_functions.pyx:6(override)
234675    0.031    0.000    0.031    0.000 cythonic/core/ant.pyx:140(set_state)
228536    0.030    0.000    0.030    0.000 cythonic/core/ant.pyx:125(out_of_bounds)
5    0.026    0.005    0.026    0.005 cythonic/core/sim_recorder.pyx:54(flush_resultset)
1    0.006    0.006    9.560    9.560 cythonic/core/sim_recorder.pyx:80(run_sim)
106    0.005    0.000    0.005    0.000 {method 'reduce' of 'numpy.ufunc' objects}
1    0.004    0.004    0.004    0.004 cythonic/plugins/db_controller.pyx:26(new_sim_id)
1    0.004    0.004    9.643    9.643 record_and_play.py:34(profiled_run)
5    0.004    0.001    0.004    0.001 {method 'copy' of 'numpy.ndarray' objects}
6019    0.003    0.000    0.017    0.000 cythonic/core/ant.pyx:118(reverse)
104    0.002    0.000    0.009    0.000 numpy/core/fromnumeric.py:69(_wrapreduction)
1    0.002    0.002    9.639    9.639 record_and_play.py:38(run)
3    0.002    0.001    0.002    0.001 {built-in method numpy.copyto}
2001    0.002    0.000    1.242    0.001 cythonic/core/sim_recorder.pyx:71(record_step)
103    0.001    0.000    0.010    0.000 numpy/core/fromnumeric.py:1966(sum)
120    0.001    0.000    0.048    0.000 cythonic/core/queen.pyx:50(generate_state)
103    0.001    0.000    0.001    0.000 {built-in method builtins.getattr}
2001    0.000    0.000    0.000    0.000 cythonic/core/ant.pyx:78(next_step)
2800    0.000    0.000    0.000    0.000 cythonic/core/domain.pyx:35(constraint)
1    0.000    0.000    0.000    0.000 cythonic/core/sim_controller.pyx:60(new_positions)
104    0.000    0.000    0.000    0.000 {method 'items' of 'dict' objects}
104    0.000    0.000    0.000    0.000 numpy/core/fromnumeric.py:70(<dictcomp>)
115    0.000    0.000    0.000    0.000 {built-in method builtins.isinstance}
103    0.000    0.000    0.678    0.007 cythonic/core/domain.pyx:106(entropy)
1    0.000    0.000    0.000    0.000 /home/bram/theANT3000/cythonic/plugins/queries.py:40(
    insert_results)
```

Appendix C

Simulation parameters

Table C-1: Experiment parameters

Part	Parameter	Symbol	scenario 1	scenario 2	scenario 3
Simulation	Steps	N		2000	
	Time step	δ_t		0.3 s	
	Start pheromone level	$w_{i,j0}$		1	
	Deploy distribution			Gamma(10,2)	
	Swarm size	M	120	120	[varying]
Agent	Speed	v		125mm/s	
	Antenna length	l		50mm	
	Stinger distance	d	25mm	10mm	10mm
	Sensor offset	Γ	45 deg	25 deg	25 deg
	Base deposition quantity	q_0		1	
	Deposition model	$q(k)$		Exponential decay	
	Exponential decay beta	γ		0.03	
	Deposition halflife time	$t_{\frac{1}{2}}$		23 s	
	Deposit return factor			2	
Control	Sensor activation	$g(w_{i,j})$	Linear (1:1)	ReLu	ReLu
	Breakpoint (ReLu)	z_{br}	-	0.25	0.25
	Control model		'simple'	'weber'	'weber'
	Gain	α	0.75	1.25	1.25
	Noise distribution		Telegraphic	Gaussian	Gaussian
	Noise gain or covariance	σ_1^2	1	0.5	0.5
	Noise covariance	σ_2^2	-	0.1	0.1
	Telegraph sojourn parameter	β	10	-	-
Domain	Pitch	Δh		10 mm	
	Size	X, Y		[2500,1000] mm	
	Nest location	x_n, y_n		[500,500] mm	
	Food location	x_f, y_f		[2000,500] mm	
	Nest, food radius	R		150 mm	
Pheromone	Evaporation rate	ρ	0.97 $\frac{1}{s}$	0.97 $\frac{1}{s}$	[varying]
	Half-life time	$t_{\frac{1}{2}}$	23 s	23 s	[varying]
	Distribution covariance		400	200	200
	Significant digits	ε		2	

List of Symbols

δ_t Inter sample time [s]

l Agent antenna length [mm]

Γ Angle between sensor location and centerline from the center of rotation [degree]

d Agent distance between center of rotation and actuator location [mm]

$\mathcal{A}$ Agents (set)

v Agent speed [mm/s]

θ Agent orientation w.r.t. the reference frame [degrees]

$\vec{\tau}$ Pheromone concentration perception as sensed with left and right antenna

ω Agent rotational speed, azimuth rate of changes [degrees/s]

$\mathcal{D}$ Domain (set)

$\mathcal{I}$ Tuple set of indexes (set)

$\vec{\Phi}$ Position vector of the agent center of rotation w.r.t. the origin of the map [mm,mm]

$\mathcal{G}$ Graph (set)

$\mathcal{N}$ Nodes (set)

$\mathcal{W}$ Weights (set)

$w_{i,j}$ Graph weights indexed by i and j [-]

λ Decay constant [1/s]

$t_{\frac{1}{2}}$ Half-life time [s]

w Pheromone quantity/graph weight [-]

ρ Pheromone evaporation rate [1/s]

α Steering gain [-]

ϵ Steering noise [-]

$\Delta\tau$ Agent pheromone contribution [-]

T_s Sojourn time, time between state changes, distributed according to $T_s \sim Exp(1/\beta)$ [s]

ς Augmented telegraph noise series [-]

Glossary

CR Center of Rotation

2D two-dimensional

`theANT3000` Simulator software package

L. humile Argentine ant *Linepithema humile* formerly known as Iridomyrmex humilis

ACO Ant Colony Optimization

TSP Traveling Salesman Problem

IR infrared

RFID Radio-frequency Identification

agent Artificial representation of an actor in a domain

IMU Inertial Measurement Unit

Bibliography

[1] lirtlon, "Army ant bridge," [Accessed May 2019]. [Online]. Available: https://glbiomimicry.org/About/

[2] P. P. Grassé, "La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs," *Insectes sociaux*, vol. 6, pp. 41–80, 1959.

[3] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[4] D. J. Ornia, "Ant Colony Algorithms and its applications to Autonomous Agents Systems," Msc Thesis, KTH Royal Institute of Technology, 2017.

[5] B. Durieux, "Literature survey," Delft University of Technology, Delft, Tech. Rep., 2019.

[6] R. Beckers, S. Goss, J.-L. Deneubourg, and J.-M. Pasteels, "Colony size, communication and ant foraging strategy," *Psyche: A Journal of Entomology*, vol. 96, no. 3-4, pp. 239–256, 1989.

[7] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: An Autocatalytic Optimizing Process Technical Report 91-016," Politecnico di Milano, Tech. Rep., feb 1991.

[8] M. A. P. Garcia, O. Montiel, O. Castillo, R. Sepúlveda, and P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," *Applied Soft Computing*, vol. 9, no. 3, pp. 1102–1110, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494609000349

[9] C. Luo, F. Shen, H. Mo, and Z. Chu, "An Improved Ant-Driven Approach to Navigation and Map Building," in *International Conference on Swarm*

Intelligence, ser. Advances in Swarm Intelligence, H. Takagi and Y. Shi, Eds. Cham: Springer International Publishing, 2017, pp. 301–309. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-61824-1_33

[10] A. Perna, B. Granovskiy, S. Garnier, S. C. Nicolis, M. Labédan, G. Theraulaz, V. Fourcassié, and D. J. T. Sumpter, "Individual Rules for Trail Pattern Formation in Argentine Ants (Linepithema humile)," *PLOS Computational Biology*, vol. 8, no. 7, p. e1002592, 2012. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1002592

[11] M. Vela-Pérez, M. A. Fontelos, and S. Garnier, "From individual to collective dynamics in Argentine ants (Linepithema humile)," *Mathematical Biosciences*, vol. 262, pp. 56–64, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0025556415000097

[12] S. Alers, K. Tuyls, B. Ranjbar-Sahraei, D. Claes, and G. Weiss, "Insect-Inspired Robot Coordination: Foraging and Coverage," in *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. Manhattan, New York: The MIT Press, jul 2014, pp. 761–768. [Online]. Available: https://www.mitpressjournals.org/doi/abs/10.1162/978-0-262-32621-6-ch123

[13] R. Fujisawa, H. Imamura, T. Hashimoto, and F. Matsuno, "Communication using pheromone field for multiple robots," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1391–1396.

[14] A. Russell, "Mobile robot guidance using a shot-lived heat trail," *Robotica*, vol. 11, no. November 1993, 1993.

[15] T. Sakakibara and D. Kurabayashi, "Artificial pheromone system using RFID for navigation of autonomous robots," *Journal of Bionic Engineering*, vol. 4, no. 4, pp. 245–253, 2007.

[16] R. Johansson and A. Saffiotti, "Navigating by stigmergy: A realization on an RFID floor for minimalistic robots," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2009, pp. 245–252.

[17] N. Lemmens, S. Alers, and K. Tuyls, "Bee-inspired foraging in a real-life autonomous robot collective," in *Proceedings of the 23rd Benelux Conference on Artificial Intelligence (BNAIC)*, Groningen, 2011, pp. 459–460. [Online]. Available: https://www.researchgate.net/publication/236950506_Bee-inspired_foraging_in_a_real-life_autonomous_robot_collective

[18] C. R. Reid, D. J. T. Sumpter, and M. Beekman, "Optimisation in a natural system: Argentine ants solve the Towers of Hanoi," *The Journal of Experimental Biology*, vol. 214, no. 1, pp. 50 LP – 58, jan 2011. [Online]. Available: http://jeb.biologists.org/content/214/1/50.abstract

[19] E. B. Melo and A. F. R. Araújo, "Modelling foraging ants in a dynamic and confined environment," *Biosystems*, vol. 104, no. 1, pp. 23–31, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030326471000239X

[20] F. Schweitzer, K. Lao, and F. Family, "Active random walkers simulate trunk trail formation by ants," *Biosystems*, vol. 41, no. 3, pp. 153–166, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030326479601670X

[21] B. Davis, "Reinforced random walk," *Probability Theory and Related Fields*, vol. 84, no. 2, pp. 203–229, 1990. [Online]. Available: https://doi.org/10.1007/BF01197845

[22] A. Stevens and H. Othmer, "Aggregation, Blowup, and Collapse: The ABC's of Taxis in Reinforced Random Walks," *SIAM Journal on Applied Mathematics*, vol. 57, no. 4, pp. 1044–1081, aug 1997. [Online]. Available: https://doi.org/10.1137/S0036139995288976

[23] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, jun 2015, pp. 1094–1099. [Online]. Available: http://ieeexplore.ieee.org/document/7225830/

[24] M. S. Blum and G. N. Ross, "Chemical releasers of social behaviour—V. Source, specificity, and properties of the odour trail pheromone of Tetramorium guineense (F.)(Formicidae: Myrmicinae)," *Journal of Insect Physiology*, vol. 11, no. 7, pp. 857–868, 1965.

[25] F. E. Regnier and J. H. Law, "Insect pheromones," *Journal of Lipid Research*, vol. 9, no. 5, pp. 541–551, 1968.

[26] I. D. Couzin and N. R. Franks, "Self-organized lane formation and optimized traffic flow in army ants," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 270, no. 1511, pp. 139 LP – 146, jan 2003. [Online]. Available: http://rspb.royalsocietypublishing.org/content/270/1511/139.abstract

[27] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant," *Journal of Insect Behavior*, vol. 3, no. 2 LB - Deneubourg1990, pp. 159–168, 1990. [Online]. Available: https://doi.org/10.1007/BF01417909

[28] V. Calenbuhr and J.-L. Deneubourg, "A model for osmotropotactic orientation (I)," *Journal of theoretical biology*, vol. 158, no. 3, pp. 359–393, 1992.

[29] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997. [Online]. Available: https://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf

[30] E. H. Weber, *De Pulsu, resorptione, auditu et tactu: Annotationes anatomicae et physiologicae...* CF Koehler, 1834.

[31] S. Aron, R. Beckers, J. L. Deneubourg, and J. M. Pasteels, "Memory and chemical communication in the orientation of two mass-recruiting ant species," *Insectes Sociaux*, vol. 40, no. 4, pp. 369–380, dec 1993. [Online]. Available: https://doi.org/10.1007/BF01253900

[32] G. Margolin and E. Barkai, "Nonergodicity of a time series obeying Lévy statistics," *Journal of statistical physics*, vol. 122, no. 1, pp. 137–167, 2006. [Online]. Available: https://arxiv.org/abs/cond-mat/0504454

[33] S. Garnier, A. Guérécheau, M. Combe, V. Fourcassié, and G. Theraulaz, "Path selection and foraging efficiency in Argentine ant transport networks," *Behavioral Ecology and Sociobiology*, vol. 63, no. 8, pp. 1167–1179, jun 2009. [Online]. Available: https://doi.org/10.1007/s00265-009-0741-6

[34] T. J. Czaczkes, "How to not get stuck—Negative feedback due to crowding maintains flexibility in ant foraging," *Journal of Theoretical Biology*, vol. 360, pp. 172–180, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022519314003981

[35] Scientific community, "Numpy random.gamma," [Accessed June 2019]. [Online]. Available: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.gamma.html

[36] B. Collignon, L. E. Cervantes Valdivieso, and C. Detrain, "Group recruitment in ants: Who is willing to lead?" *Behavioural Processes*, vol. 108, pp. 98–104, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0376635714002381

[37] Gc-tronic, "Elisa-3," [Accessed March 2019]: http://www.gctronic.com/doc/index.php/Elisa-3. [Online]. Available: http://www.gctronic.com/doc/index.php/Elisa-3

[38] C. Grüter, D. Maitre, A. Blakey, R. Cole, and F. L. W. Ratnieks, "Collective decision making in a heterogeneous environment: Lasius niger colonies preferentially forage at easy to learn locations," *Animal Behaviour*, vol. 104, pp. 189–195, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003347215001177

[39] O. Feinerman and J. F. A. Traniello, "Social complexity, diet, and brain evolution: modeling the effects of colony size, worker size, brain size, and foraging behavior on colony fitness in ants," *Behavioral Ecology and Sociobiology*, vol. 70, no. 7, pp. 1063–1074, jul 2016. [Online]. Available: http://link.springer.com/10.1007/s00265-015-2035-5

[40] M. D. Breed and J. Moore, "Chapter 14 - Comparative Social Behavior," in *Animal Behavior (Second Edition)*, 2nd ed., M. D. Breed and J. Moore, Eds. San Diego: Academic Press, 2016, pp. 459–497. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128015322000143

[41] Prodrive, "Manufacturing solutions," [Accessed June 2019]. [Online]. Available: https://prodrive-technologies.com/solutions/manufacturing-solutions/

[42] E. O. Wilson, "The social biology of ants," *Annual Review of Entomology*, vol. 8, no. 1, pp. 345–368, 1963.

[43] Z. Cook, D. W. Franks, and E. J. H. Robinson, "Exploration versus exploitation in polydomous ant colonies," *Journal of Theoretical Biology*, vol. 323, pp. 49–56, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022519313000477

[44] B. Durieux, "Github - Elmosnewshoes," 2019. [Online]. Available: https://github.com/Elmosnewshoes/Stigmergy/

[45] K. W. Smith, *Cython: A Guide for Python Programmers*, 1st ed. Sebastopol, United States: O'Reilly Media, 2015.