INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Use of Bio-inspired Techniques to Solve Complex Engineering Problems: Industrial Automation Case Study

José Fernando Lopes Barbosa

Relatório de Projecto para obtenção do grau de Mestre em

Engenharia Industrial

ramo de especialização em Engenharia Electrotécnica

Supervision: Prof. Dr. Paulo Leitão

September, 2010

Dedication

To Inês and Mariana

Acknowledgements

First of all I want to express my most grateful thanks to Professor Paulo Leitão for his knowledge, patient, support and constant motivation during the development of this work without which the conclusion was not possible.

Next, I want to thank all my friends for their support, incentive and motivation.

I also want to thank my parents for all the support and love throughout the years.

A special thank to my beautiful grandmother that with her love and affection made me feel to be her special grandson.

Finally, but not the least, I want to thank my wife Inês for her support and comprehension and to my daughter Mariana just to make me constantly smile.

Abstract

Nowadays local markets have disappeared and the world lives in a global economy. Due to this reality, every company virtually competes with all others companies in the world. In addition to this, markets constantly search products with higher quality at lower costs, with high customization. Also, products tend to have a shorter period of life, making the demanding more intense. With this scenario, companies, to remain competitive, must constantly adapt themselves to the market changes, i.e., companies must exhibit a great degree of self-organization and self-adaptation.

Biology with the millions of years of evolution may offer inspiration to develop new algorithms, methods and techniques to solve real complex problems. As an example, the behaviour of ants and bees, have inspired researchers in the pursuit of solutions to solve complex and evolvable engineering problems.

This dissertation has the goal of explore the world of bio-inspired engineering. This is done by studying some of the bio-inspired solutions and searching for bio-inspired solutions to solve the daily problems. A more deep focus will be made to the engineering problems and particularly to the manufacturing domain.

Multi-agent systems is a concept aligned with the bio-inspired principles offering a new approach to develop solutions that exhibit robustness, flexibility, responsiveness and re-configurability. In such distributed bio-inspired systems, the behaviour of each entity follows simple few rules, but the overall emergent behaviour is very complex to understand and to demonstrate. Therefore, the design and simulation of distributed agent-based solutions, and particularly those exhibiting self-organizing, are usually a hard task. Agent Based Modelling (ABM) tools simplifies this task by providing an environment for programming, modelling and simulating agent-based solutions, aiming to test and compare alternative model configurations. A deeply analysis of the existing ABM tools was also performed aiming to select the platform to be used in this work.

Aiming to demonstrate the benefits of bio-inspired techniques for the industrial automation domain, a production system was used as case study for the development of a self-organizing agent-based system developed using the NetLogo tool.

Keywords: Bio-inspired, Self-organization, Automation, NetLogo

Resumo

Hoje em dia os mercados locais desapareceram e o mundo vive numa economia global. Devido a esta realidade, cada companhia compete, virtualmente, com todas as outras companhias do mundo. A acrescentar a isto, os mercados estão constantemente à procura de produtos com maior qualidade a preços mais baixos e com um grande nível de customização Também, os produtos tendem a ter um tempo curto de vida, fazendo com que a procura seja mais intensa. Com este cenário, as companhias, para permanecer competitivas, têm que se adaptar constantemente de acordo com as mudanças de mercado, i.e., as companhias têm que exibir um alto grau de auto-organização e auto-adaptação.

A biologia com os milhões de anos de evolução, pode oferecer inspiração para desenvolver novos algoritmos, métodos e técnicas para resolver problemas complexos reais. Como por exemplo, o comportamento das formigas e das abelhas inspiraram investigadores na descoberta de soluções para resolver problemas complexos e evolutivos de engenharia.

Esta dissertação tem como objectivo explorar o mundo da engenharia bio-inspirada. Isto é feito através do estudo de algumas das soluções bio-inspiradas existentes e da procura de soluções bio-inspiradas para resolver os problemas do dia-a-dia. Uma atenção especial vai ser dada aos problemas de engenharia e particularmente aos problemas do domínio da manufactura.

Os sistemas multi-agentes são um conceito que estão em linha com os princípios bio-inspirados oferecendo uma abordagem nova para desenvolver soluções que exibam robustez, flexibilidade, rapidez de resposta e reconfiguração. Nestes sistemas distribuídos bio-inspirados, o comportamento de cada entidade segue um pequeno conjunto de regras simples, mas o comportamento emergente global é muito complexo de perceber e de demonstrar. Por isso, o desenho e simulação de soluções distribuídas de agentes, e particularmente aqueles que exibem auto-organização, são normalmente uma tarefa árdua. As ferramentas de Modelação Baseada de Agentes (MBA) simplificam esta tarefa providenciando um ambiente para programar, modelar e simular, com o objectivo de testar e comparar diferentes configurações do modelo. Uma análise mais aprofundada das ferramentas MBA foi também efectuada tendo como objectivo seleccionar a plataforma a usar neste trabalho.

Com o objectivo de demonstrar os benefícios das técnicas bio-inspiradas para o domínio da automação industrial, um sistema de produção foi usado como caso de estudo para o desenvolvimento, usando a ferramenta NetLogo, de um sistema baseado em agentes auto-organizado.

Palavras-chave: Bio-inspiração, Auto-organização, Automação, NetLogo

Acronyms

ABM	Agent Based Modelling
ACO	Ant Colony Optimization
ADACOR	ADAptive holonic Control aRchitecture for distributed manufacturing systems
AIS	Artificial Immune Systems
API	Application Programming Interfaces
DAS	Dynamic Assembly System
GA	Genetic Algorithm
GUI	Graphical User Interface
IDE	Integrated Development Environment
MAS	Multi Agent System
MLT	Manufacturing Lead Time
NP	Non-deterministic Polynomial-time hard
PSO	Particle Swarm Optimization
RFID	Radio Frequency IDentification
WIP	Work In Process

Contents

List of Figures

List of Tables

1. Introduction

The solutions for complex problems are usually found where are less expected, being necessary to open our eyes and look for successful cases in our living days. In biology and nature the systems are complex and adaptive, but the individual entities are very simple and with very limited cognitive skills. In such systems, the system behaviour is based on simple and adaptive individuals that cooperate with each other in order achieve the whole objective. The biology ideas, mainly swarm intelligence and self-organization, have been the source of inspiration for the development of several techniques and methods to solve complex engineering problems. Problems like logistics and traffic optimization, telecommunications networks, economic markets and production systems have bio-inspired solutions [Leitão, 2009b]. Particularly, the application of techniques inspired in biology can contribute to achieve manufacturing systems with the desired characteristics of robustness, flexibility and re-organization.

With simple rules to coordinate the global behaviour, the required software to develop agent-based solutions is shorter and simpler than the software required by the centralized approaches, leading to easier development, debug and maintenance [Parunak, 1996]. However, the development and debug of agent-based systems remain a difficult task, especially when these distributed systems exhibit complex phenomena, such as emergence and self-organization. Additionally, the simulation of these systems at the design phase, allowing testing different control strategies, the tuning configuration parameters and identifying mistakes and misunderstanding, brings important benefits if they are done before the deployment into the practical operation.

1.1 Motivation and Objectives

The motivation of this work is to understand how bio-inspired techniques can be used to solve complex engineering problems. For this purpose, several bio-inspired techniques will be studied and existing applications will be surveyed, analysing the use of different bio-inspired methods to different application domains. A particular attention will be given to the manufacturing and automation fields, discussing the applicability of such bio-inspired solutions to the different areas within the manufacturing domain.

In this work, several Agent Based Modelling (ABM) tools, which allows the modelling and simulation of bio-inspired agent-based solutions, will be studied and compared. Due to their nature, ABM tools are a powerful way to validate ideas allowing the fast prototyping and proof-of-concept, i.e. the creation, simulation and/or validation of the agents' behaviour for a desired problem in an easy way.

In the perspective to demonstrate the potential of bio-inspired algorithms to offer good alternative solutions, a production system case study will be used to develop an agent-based control system that exhibits self-organization capabilities. For this purpose, it will be used the NetLogo platform to model and simulate the system behaviour using different control algorithms running under different scenarios.

1.2 Dissertation organization

This document is organized in seven chapters, starting with the present chapter where the contextualization, problem and objectives were presented.

The second chapter, entitled "Bio-inspired Techniques, Algorithms and Methods", presents an overview of some nature behaviours that are mimic and makes the bridge to the implementation of the bio-inspired algorithms.

The chapter 3, entitled "Survey of Applications of Bio-inspired Solutions", surveys the applications of the bio-inspired techniques to solve daily and engineering problems and in particular complex problems found in automation/manufacturing world.

The fourth chapter, entitled "Simulations Tools to Support Bio-inspired Engineering", discusses the use of Agent Based Modelling tools, analyzing and comparing several existing tools and detailing the NetLogo environment that will be used in this work to develop the self-organized agent-based solution.

The chapter 5, entitled "Self-organizing Agent-based Model for an Automation System", describes the case study selected to illustrate the application of bio-inspired techniques in automation systems, and provides details about the implementation of the agent-based model using the NetLogo environment.

In chapter 6, entitled "Analysis of the Experimental Results", the developed agent-based model is simulated under several scenarios, and the experimental results are analyzed allowing to reach important conclusions about the importance of these bio-

inspired techniques to improve the engineering of modular and reconfigurable automation solutions.

Finally, the last chapter rounds up the document with the conclusions and points out some future work.

2. Bio-inspired Techniques, Algorithms and Methods

Planet Earth was born about 4.6 Billion years ago and carries with that the same amount of time of life creation, refining and evolution. Because of that, nature has millions of species and inherent to that, nature has a lot of powerful mechanisms to offer, to handle emergent and evolvable environments [Leitão, 2009b]. And the beauty of this evolution is that it comes with the particularity that the species have simple ruling mechanisms that govern them and even so very complex behaviours can emerge.

Other situation observed in nature and that can be very useful to solve a large number of problems is that some species have a well defined distribution of jobs and each individual knows what to do (i.e. division of labour). The study of these species can give some insights to problem solving

Humans are also studying certain parts of the human body and are trying to understand their behaviour and their applicability in real life problem solving. One example is the human Immune System that has given birth to some bio-inspired algorithms [Castro and Timmis, 2002]. Other example can be found on the functioning of the human brain that also gave origin to the Neural Networks [Fausett, 1994].

This chapter gives a generalized view of some mechanisms found in biology that can be copied to solve complex engineering problems, and introduces some bio-inspired techniques, algorithms and methods. This chapter doesn't intent to give an exhaustive description of all existent bio-inspired techniques, but intends to make a brief overview of the most known and used, mainly the swarm intelligence and self-organization.

2.1 Swarm Intelligence

Swarm Intelligence is a concept found in colonies of insects that can be defined as "the emergent collective intelligence of groups of simple and single entities" [Bonabeau et al., 1999]. Swarm intelligence offers an alternative way of designing intelligent, complex systems. In these complex systems the traditional centralized control is replaced by a distributed functioning where the interactions between individuals leads to the emergence of "intelligent" global behaviour unknown to them [Bonabeau et al., 1999]. Some examples of swarm intelligence include ant colonies, bird flocking, fish shoaling and bacterial growth [Miller, 2007].

2.1.1 Biological Concept

In nature and biology, complex systems are built upon entities that exhibit simple behaviours, made of a small set of simple rules, and having reduced cognitive abilities. In fact the behaviour of the whole emerges from the contribution and the interaction of every single entity, i.e., the collective behaviour. These complex systems don't have a pre-defined master plan or one central entity. Instead of that, the behaviour of the system is decentralized. The behaviour of the whole is greater and much more complex than the sum of the single behaviours [Holland, 1999]. A well-known example is the movement of group of birds, where individuals coordinate their movements according to the movement of the others (e.g. the typical V formation).

Figure 1 - V formation shape

Other examples of species that exhibit these characteristics are ants and honey bees. It is common sense knowledge that this two species are not very smart, but their colonies are, and concretely the emergent behaviours of a society of ants or bees are very surprisingly complex [Miller, 2007].

Figure 2 - Bird flocking

Simple mechanisms are used to coordinate the individual behaviours of these entities, namely feedback mechanisms. Feedback mechanisms use positive and negative

indications to regulate the system behaviour. i) in case of positive feedback, the system responds to the perturbation in the same direction as the change, towards amplification, and ii) in case of negative feedback, the system responds to the perturbation in the opposite direction, towards stabilization. Combining positive and negative feedbacks, the system can be maintained under control but pushed to its limits [Camazine et al., 2001]. In fish nesting, the coordination process uses a simple rule *"I nest where other similar individuals nest unless there are too much fishes"*, where the first part is related to the positive feedback, allowing to increase the aggregation of fishes at the same place, and the second part is related to the negative feedback, avoiding a great concentration of fishes at the same place (see Figure 3).

Figure 3 - Fish schooling (avoidance of predator)

Other similar mechanisms, found in other areas of science, are the market laws [Márkus et al., 1996] and potential (attraction) fields [Vaario and Ueda, 1996], which use the concept of regulating expectations of individuals presenting conflict of interests: e.g. some entities have operations to be executed and others have skills to execute them.

In these colonies, each individual entity only possesses a very limited view of the surrounding world, but a larger view is needed and because of this larger view a communication mechanism must be achieved. Instead of using direct communication, it is usual the use of indirect communication, mainly through the environment. In case of ants, the communication is achieved by using a spreading odour chemical substance known as pheromone. This indirect communication mechanism is achieved by the deposit of pheromones in the paths to food sites. In the presence of a pheromone trail, other ants know that they are in presence of a path to a food site. Each ant reinforces the pheromone trail telling this way that the path is a valid one.

Pheromones in the nature suffer from a natural process of reduction of intensity. This odour reduction is as higher as the time elapsed from the nest to the food source, i.e. as longer is the travelled distance. If several ants make different trips to the same food source, several paths to the same solution will appear. The optimal solution will be the shortest one, i.e. that whose pheromones have a more intense odour. In fact, the double bridge experience, conducted by [Deneubourg et al., 1990] states that, for several experiences, on the presence of two equal paths from a nest to a food source each path is chosen 50% of the times and that in each experience ants tend to chose only one path. If by other hand one path is significantly bigger than the other, ants chose the shortest one [Goss et al., 1989].

Other illustrative example of swarm intelligence is related to the waggle dance used by honey bees to exchange information about the direction and distance to patches of flowers yielding nectar and pollen, Figure 4.

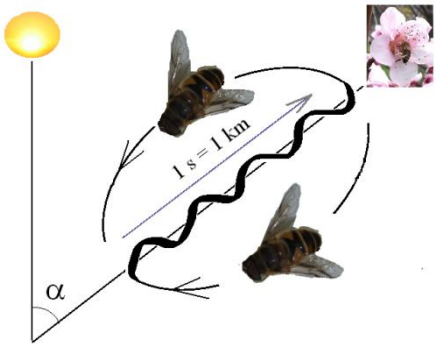

Figure 4 – Bee waggle dance

In the waggle dance the angle from the sun indicates the direction and the duration of the waggle part of the dance represents the distance to the patch [Frisch, 1967].

2.1.2 Ant Colony Optimization

Ant colony optimization (ACO) is a meta-heuristic algorithm that is inspired on the behaviour of food foraging of ants. It was first introduced in 1992 by Marco Dorigo in his PhD thesis [Dorigo, 1992]. The inspiration on the algorithm's development came from the fact that real ants have the ability to find the shortest path from source food to the nest and also the flexibility to find new ones when an obstacle arises on the previous path [Beckers et al., 1992].

Basically, the ants (or software agents) travel over a weighted graph in a random manner, depositing on the way a trail (pheromones). After a period of time, when the pheromone level is considerable, the ants leave to travel in a random manner and start to follow the trail with the most intense level of pheromone.

In a generic view, the algorithm can be implemented following the next adapted pseudo code from [Dorigo, 2007].

```
Set parameters, initialize pheromone trails
While conditions not met do
    ConstructAntSolutions
    DaemonActions  {optional}
    UpdatePheromones
End While
```

The ACO algorithm and its variations have been successfully applied to solve NP-hard and dynamic NP-hard problems [Dorigo and Stützle, 2009]. Dynamic NP-hard problems have an increased degree of difficulty which is related to the fact that variables can be time-varying stochastic variables, i.e., change over time. A short list of applications, which will be more detailed in chapter 3, includes routing, assignment, scheduling, machine learning and bioinformatics problems.

2.1.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an algorithm inspired on the behaviour of fish schooling and bird flocking. The algorithm is a population based stochastic optimization technique that was first introduced by [Eberhart and Kennedy, 1995].

The system is initialized with a population of random solutions and searches for the optimal one by updating generations. The potential solutions (or particles) fly through the problem space by following the current optimum particles. As the swarm iterates, the fitness of the global best solution is improved (decreases for minimization problem).

A pseudo code of the algorithms found in [Hu, 2010] is the following.

```
For each particle
    Initialize the particle
End For
Do
```

```
For each particle
  Calculate fitness value
  If the fitness value is better than the best fitness
    value (pBest) in history
     Set current value as the new pBest
End For
Choose the particle with the best fitness value of all the
  particles as the gBest
For each particle
  Calculate particle velocity
  Update particle position
End For
While maximum iterations or minimum error criteria is not
  attained
```

The list of applications, which will be more detailed in chapter 3, includes areas related to image and video, biomedical, communication networks, electronics, power systems and robotics.

2.1.4 Bees Algorithm

In a food foraging situation, honey bees start by sending scout bees in the search of good flower patches. After they return to the hive, and if a good food source was found, scout bees inform others by doing a dance, know as waggle dance [Bonabeau et al., 1999, Frisch, 1967]. This waggle dance provides the following information: the direction, the distance and food quality of the site. After this waggle dance, bees follow the scout bee to the food site. While harvesting, bees measure the amount of food left, to see if that particular food site is still valid or not, being either way advertised in hive with the waggle dance.

This behaviour was mimic to an algorithm by Prof. D.T. Pham and his colleagues [Pham et al., 2005] and has, in its basic form, the following pseudo code:

```
Initialise population with random solutions.
Evaluate fitness of the population.
While (stopping criterion not met)
    Select sites for neighbourhood search.
    Recruit bees for selected sites (more bees for best e
      sites) and evaluate fitnesses.
    Select the fittest bee from each patch.
    Assign remaining bees to search randomly and evaluate their
      fitnesses.
End While.
```

The list of applications, which will be more detailed in chapter 3, includes scheduling, data clustering, design and manufacturing.

2.2 Evolution and Self-organization

Species tend to adapt to better suit their environment. Changes occur from generation to generation that tend to eliminate their limitations and even with very small changes over time they produce major results.

This chapter will give an insight to algorithms that have inherited characteristics of evolution and of self-organization.

2.2.1 Biological Concepts

The Darwinian theory of evolution is a form for the adaptation to the dynamic evolution of the environment. According to Darwin, nature is in a state of permanent transformation, in which the species change from generation to generation, evolving to better adapt to their environment. Darwin saw the evolution as a result of selection by the environment acting on a population of organisms competing for resources. In this evolution process, the selection is natural in the sense that is purely spontaneous without a pre-defined plan.

Another approach to evolution is the concept of self-organization. Several distinct definitions, but not necessarily contradictory, are found in the literature [Bousbia and D. Trentesaux, 2002, Massotte, 1995, Vaario and Ueda, 1996], but a possible definition to be used in this work can be: "*The ability of an entity/system to adapt dynamically its behaviour to external changing conditions without external intervention"* [Leitao, 2008].

Self-organization occurs when species without a predefined plan or one entity in charge adapt to external changes. One example of the usage of self-organization properties is the mound building and reshape by termites. During building, termites use signs to inform others of what to do. In the same manner, if the external conditions change, e.g. the direction of wind, termites re-arrange their mound in order to obtain the desired wind entry.

Self-organizing systems don't have a rigid and estimated organization, evolving through a non-linear and dynamic process with a constant optimization of the individuals' behaviour.

The utilization of self-organization allows the achievement of self-* properties, namely:

- *Self-configuration*, that is the capacity to dynamically adapt to when conditions change, by modifying its own configuration allowing the addition/removal of resources on the fly (i.e. without the need to stop, re-program and start the other components) and guarantying service disruption.
- *Self-optimization*, that is the capacity to adjust itself in a pro-active way to respond to environmental stimulations.
- *Self-healing*, that is the capacity to sense deviations from non optimal conditions and take proactive actions to re-establish them and avoid service disruptions. This also includes, in a more advanced view, the ability to self-repair.

Several examples of self-organization can be found in nature, such as stigmergy and thermodynamics (decrease of entropy) and autopoiesis [Leitão, 2009b]

2.2.2 Genetic Algorithms

The Genetic Algorithms (GA) is based on a population of abstract representations of candidate solutions to an optimization problem that evolves toward better solutions. GA applies evolution operators, namely inheritance, mutation, selection, and crossover.

Like the others algorithms (e.g. ACO and PSO) there are different variations of GA but the basic algorithm could have the following form: a population is created randomly with a group of individuals. Then, these individuals are evaluated by an evaluation function provided by the programmer. This evaluation function acts like a filter to select the most fittest to reproduce. The selection is based on the fitness that each individual gets, which is done for two individuals. After the creation of one or more offspring the individuals are randomly mutated. This cycle continues until a good solution is found or a maximum number of generations are reached. The pseudo code of this implementation found at [Skinner, 2010] is reproduced here.

```
For all members of population
    sum += fitness of this individual
```

```
End for

For all members of population
    probability = sum of probabilities + (fitness / sum)
    sum of probabilities += probability
End for

Loop until new population is full
     Do this twice
       number = Random between 0 and 1
       For all members of population
           If number > probability but less than next
              probability
                 Then you have been selected
       End for
     End
     Create offspring
End loop
```

GA areas of application, to be more detailed in chapter 3, include astronomy and astrophysics, electrical engineering, robotics, routing and scheduling and systems engineering.

2.2.3 Stigmergy

Stigmergy is a form of self-organization, being the term Stigmergy first introduced in 1959 by the French biologist Pierre-Paul Grassé [Grassé., 1959]. The term derives from the Greek words *stigma* which means mark or sign and the word *ergon* which means work or action.

Stigmergy can be described as the deposition of signs in the environment that other entities sense to achieve/make a determined action. This indirect communication mechanism can be observed in the behaviour of social insects like termites or ants. For example, in social insects' behaviour, the phenomenon involving an indirect coordination between entities, where the trace left in the environment stimulates the execution of a subsequent action, by the same or different entity is known as stigmergy.

Figure 5 exemplifies the stigmergy mechanism. The left ant deposit the pheromone (i.e. a chemical substance that has a distinctive odour) in the ground and the following ant while walking senses that odour being guided in that way.

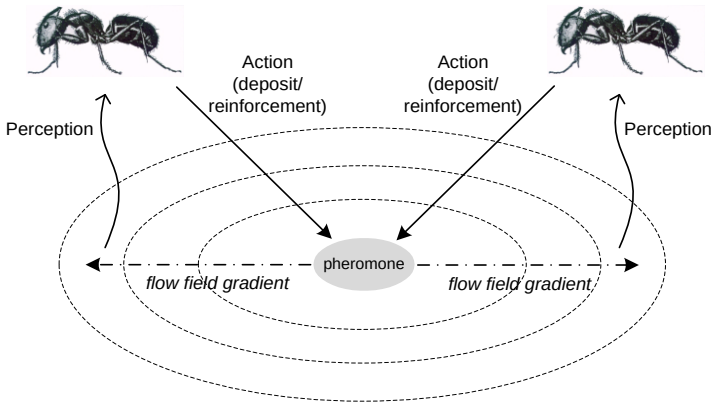

Figure 5 – Indirect Communication Among Ants [Leitão, 2009b]

According to [Parunak, 2005] stigmergic mechanisms have a good number of attractive features for military propose. These features also could be good in other applications fields and they are: simplicity, scalability, robustness and environmental integration.

Stigmergic based algorithms have several applications. Among them are military, robotics, multi-agent systems and communication in computer networks.

2.3 Artificial Immune System

Artificial Immune Systems (AIS) has its roots on the early work of [Farmer et al., 1986], and mimics the principles and processes found on the vertebrate immune system. Typically these algorithms use the characteristics of memory and learning that are exhibited.

There are several variations of AIS, but the most usual are the Negative Selection, Clonal Selection and Immune Networks [Timmis et al., 2008]. As an example, the pseudo code of a basic Negative Selection algorithm is presented. The algorithm is inspired by the main mechanism in the thymus that produces a set of mature T-cells.

```
input   : S_seen = set of seen known self elements
output  : D = set of generated detectors
Begin
   Repeat
      Randomly generate potential detectors and place them in a
        set P
      Determine the affinity of each member of P with each member
        of the self set S_seen
      If at least one element in S recognizes a detector in P
        according to a recognition threshold,
```

```
            Then the detector is rejected, otherwise it is added
              to the set of available detectors D
  Until  Stopping criteria has been met
End
```

Some of the main application areas for AIS are: Clustering/classification, anomaly detection, robotics, scheduling, fault diagnosis and numeric function optimisation.

3. Survey of Applications of Bio-inspired Solutions

Several research groups have been working in copying/adapting the previous described biological adaptive behaviour to solve complex engineering and mathematical problems. In this chapter, the application of bio-inspired techniques and methods in engineering is briefly surveyed. A special attention will be made to their applicability to the manufacturing domain.

3.1 Existing Applications to solve Mathematical/Engineering Problems

The behaviour found in ants and particularly the ACO algorithm has been applied in a wide of problem domains.

The main area of application of the behaviour of ants is probably the planning and scheduling and on this area and in a real world context, Air Liquide uses an ant-based strategy to manage the truck routes for delivering industrial and medical gases [Miller, 2007] and also [Bell and McMullen, 2004] uses it to optimize logistics vehicle routing. Another real life example is on the utilization of an ant behaviour model to improve airlines scheduling in the Sky Harbor International Airport in Phoenix [Miller, 2007].

The ACO technique is used to determine the optimal values for the components in an electronics power circuit [Zhang et al., 2008], and to achieve an optimal image thresholding, separating the object from its background [Malisia and Tizhoosh, 2006]. A solution based on the ants behaviour is applied to update dynamically and in adaptive manner the routing tables in telecommunications [Di Caro and Dorigo, 1998]. Amongst other applications are the reduction of energy consumption in remote sensor networks [Camilo et al., 2006]), the cooperation among robots (swarm robots) to achieve a complex task [Nouyan et al., 2009]. This method is also being applied in the army for the dynamic re-planning of Uninhabited Aerial Vehicles [Duan et al., 2009] and in the financial markets for the prediction of the price share process [Fang and Bai, 2009].

Another bio-inspired technique is the PSO. Briefly, PSO is being applied to solve problems that go from social to medical and from mathematical to engineering fields. As examples of such applications are the parameters optimization in the design of PID controllers [Gaing, 2004], the credit risk assessment in financial area [Li and Pi, 2009], in the design of evolvable hardware [Peña et al., 2006], in vehicle routing with simultaneous pickup and delivery [Ai and Kachitvichyanukul, 2009], and to optimize

parameters on spatiotemporal retina models [Niu et al., 2007]. A more extensive analysis and detailed classification of applications of PSO can be found in [Poli, 2007].

Other examples of application of swarm intelligence principles are in forecasting energy demands in Turkey [Miller, 2007] and in traffic and transportation problems [Teodorovic, 2008]. A more widespread example of the application of the swarm intelligence principles is Wikipedia [Leitão, 2009b] where a huge number of people contribute for the constant evolution of the encyclopaedia with their individual knowledge; no single person knows everything but collectively it is possible to know far more than it was expected to know.

GA is being successfully applied in different application domains, notably in power distribution [Ramirez-Rosado and Bernal-Agustin, 1998], in image segmentation [Peng et al., 2000] and in the military field with route scheduling and selection for land moves [Montana et al., 1999].

Table 1 summarizes the survey of applying bio-inspiration techniques to different domains.

Table 1 - Summary of bio-inpired applications

Problem domain	Existing ACO inspired solutions	Existing PSO inspired solutions	Existing GA inspired solutions
Communication Networks	[Di Caro and Dorigo, 1998] [Zhao et al., 2009] [Sim and Sun, 2002]	[Dongming et al., 2008] [Li et al., 2008]	[Lima et al., 2007] [Lee et al., 1997]
Control	[Van Ast et al., 2009] [Boubertakh et al., 2009] [Zhang and hai Wang, 2008]	[Gaing, 2004] [Jalilvand et al., 2008] [Hu et al., 2005]	[Wai and Su, 2006] [Toderici et al., 2010] [Bae et al., 2001]
Financial	[Fang and Bai, 2009] [Yuan and Zou, 2009] [Hong et al., 2007]	[Li and Pi, 2009] [Majhi et al., 2008] [Chen et al., 2009]	[Badawy et al., 2005]
Hardware design	[Zhang et al., 2008] [Abd-El-Barr et al., 2003] [Sethuram and Parashar, 2006]	[Peña et al., 2006] [Goudos et al., 2008] [Ren and Cheng, 2009]	[Tsai and Chou, 2006] [Regue et al., 2001]
Image Processing	[Malisia and Tizhoosh, 2006] [Tian et al., 2008] [Wang et al., 2005]	[Chen et al., 2009] [Chandramouli and Izquierdo, 2006] [Ma et al., 2008]	[Peng et al., 2000] [Katayama et al., 2006]
Medical	[Meng, 2006] [Lee et al., 2009] [Logeswari and	[Niu et al., 2007] [Meng, 2006]	[Maulik, 2009] [Das and

	Karnan, 2010]		Bhattacharya, 2009] [Tohka et al., 2007]
Military	[Duan et al., 2009] [Cheng et al., 2009] [Munirajan et al., 2004]	[Matlock et al., 2009] [Cui and Potok, 2007] [Thangaraj et al., 2009]	[Moore and Sinclair, 1999] [Montana et al., 1999] [Liu et al., 2005]
Power Energy	[Lee and Vlachogiannis, 2005] [Liu et al., 2009] [Colson et al., 2009]	[Liu and Ge, 2008] [Zhang et al., 2008] [Leeton et al., 2010]	[Ramirez-Rosado and Bernal-Agustin, 1998]
Robotics	[Nouyan et al., 2009]	[Zhengxiong and Xinsheng, 2010]	[Tohka et al., 2007] [Karlra and Prakash, 2003] [Pessin et al., 2009] [Albert et al., 2009]
Sensors / Sensor networks	[Camilo et al., 2006] [Muraleedharan and Osadciw, 2009]	[Aziz et al., 2007] [Tewolde et al., 2008] [Li and Lei, 2009]	[Jiang et al., 2009] [Brown and McShane, 2004] [Khanna et al., 2006]
Vehicle Routing / Traffic Control	[Miller, 2007] [Bell and McMullen, 2004]	[Ai and Kachitvichyanukul, 2009] [Wu and Tan, 2009]	[Tong et al., 2004] [Jun, 2009] [Tunjongsirigul and Pongchairerks, 2010]

3.2 Existing Applications to solve Manufacturing and Automation Problems

Manufacturing and automation domains cover a wide range of application domains presenting different requirements and constraints. From the previous examples it is clear that the existing bio-inspired solutions focus high-levels of control.

In manufacturing domain, the ACO algorithm or the ants behaviour were used in machine layouts optimization [Corry and Kozan, 2004], in shop scheduling [Blum and Sampels, 2004], and in coordination of adaptive manufacturing control systems [Hadeli et al., 2004]. In a real world application a solution based in the ACO algorithm is used on the scheduling of continuous casting aluminium in a factory located in Quebec [Gravel et al., 2002]

Algorithms based on the behaviour of honey bees has also inspired researchers to solve job scheduling problems [Pham et al., 2007] and to optimize the manufacturing layout formation [Pham et al., 2007].

PSO has been applied to detect machinery faults [Samanta and Nataraj, 2009], for job shop scheduling [Xia and Wu, 2005] and optimisation of manufacturing cells layouts and allocation of transport robots [Yamada et al., 2003].

AIS has also inspired the resolution of manufacturing problem like scheduling [Hong, 2009] [Mori et al., 1998] and also, for example, to layout optimization [Satheesh Kumar et al., 2009].

Inspiration drawn of self-organization has also being used to solve complex adaptive problems, namely in holonic manufacturing control systems [Leitão and Restivo, 2006], in the dynamic resource allocation of a factory plant of Daimler Chrysler [Bussmann and Sieverding, 2001], in the design and implementation of self-organized and self-assembled biologically inspired robots [Moudada et al., 2004] and in manufacturing scheduling [Thamarajah, 1998].

Bio-inspiration combined with holonic concepts is being used to design intelligent and adaptive manufacturing control systems (see [Leitão, 2009a]). Also in robotics, such solutions are used to design bio-inspired morphologies, sensors and actuators, and control architectures.

GA has also a word to say in this area. See for example its application in a job-shop scheduling problem [Qiu et al., 2009] and to determine optimized layouts [Wang et al., 2008].

Table 2 summarizes the application of bio-inspiration techniques to different domains.

Table 2 - Summary of bio-inspired applications to manufacturing domain

Problem domain	Existing ACO inspired solutions	Existing PSO inspired solutions	Existing GA inspired solutions
Assembly/disassembly	[Shan et al., 2007] [Sharma et al., 2009] [Lu et al., 2008]	[Lv and Lu, 2009] [Dong et al., 2007]	[Lazzerini et al., 1999] [Gao and Chen, 2008]
Layout Optimization	[Jain and Sharma, 2005] [Sun and Teng, 2002] [Chen and Rogers, 2009]	[Ning et al., 2004] [Ohmori et al., 2010] [Lei et al., 2003]	[Wang et al., 2008] [Kulkarni and Shanker, 2007]

Scheduling	[Arnaout et al., 2008] [Chen et al., 2008] [Xu et al., 2009]	[Shi et al., 2009] [Zhang and Wu, 2008]	[Qiu et al., 2009] [Aggoune et al., 2001]
Supply chain	[Suva et al., 2004] [Sun et al., 2008] [Caldeira et al., 2007]	[Sinha et al., 2009] [Qi et al., 2008]	[Elmahi et al., 2004] [Kaijun et al., 2010] [Jianhua and Xianfeng, 2010]

3.3 Challenges and Contribution Areas

In spite of the promising perspective that the bio-inspired principles can bring to engineering systems and particularly to the manufacturing domain, their adoption remains less effective than expected, mainly in industrial solutions. In fact, a major problem is the demand of industry for proven technology. The companies don't want to be the first ones to try these methods in their production processes. This requires the maturity of the technology and the proofs of its real applicability and merits. Additionally, industry has afraid of the usage of emergent terminology usually associated to these new technologies, like ontologies, self-organization, emergence, distributed thinking and learning [Leitão, 2009a].

The challenge faced to the engineer that is developing/researching bio-inspired solutions for manufacturing is to convince people of the real advantages of applying distributed behaviour based on simple, effective and adaptive entities regulated by simple coordination mechanisms as it occurs in nature. For this purpose, it is important the development of demonstrators and real case studies to be used as the proof of concept.

With this in mind, these bio-inspired techniques and methods, and especially those supporting swarm intelligence and self-organization, could have a great impact to design more intelligent, modular, flexible and adaptive systems in the following manufacturing areas:

- *Supply chains and virtual organizations*, which requires the frequent re-organization of partners aiming to achieve optimization and responsiveness to unexpected situations.
- *Shop floor (factory) layout*, where the optimization of the shop floor layout is crucial to achieve a minimization of transport operations; additionally, it is also

important where the manufacturing resources present in the shop floor are movable, i.e. the producer and transporter resources move physically in order to minimize the transportation distances.

- *Product demand*, where the manufacturing system re-organizes itself in order to adapt to the changes in the product demand, increasing or reducing the number of manufacturing resources, or modifying their capabilities, based on the forecasted production demands.
- *Planning and scheduling*, where the goal is to find optimized planning and scheduling plans taking into consideration the product demands and the capabilities of the shop floor resources.
- *Adaptive control*, where the goal is to find out an adaptive and dynamic production control strategy based in the dynamic and on-line schedule, adapted in case of occurrence of unexpected disturbances.
- *Predictive maintenance*, where the prediction of machinery failures is crucial to support disturbances and malfunctions contributing for an adaptive production system.
- *Adaptive processes and equipments*, where the development of new sensors, actuators and controllers will contribute to design and implement more adaptive manufacturing equipments.

Note that such solutions may be as more useful as more unpredictable will be the environment where they run. Also, bio-inspired concepts are more suited, at least for yet, to a higher level of control since they have lower requirements of real time implementation.

4. Simulations Tools to Support Bio-inspired Engineering

An agent is an entity that has built upon a set of behaviour rules exhibiting some properties, such as autonomy and cooperation. Each agent has a local view of the surrounding world (i.e. it doesn't have a global knowledge) and a decision cannot be achieved by a simple agent. Each agent is autonomous but can, if necessary, communicate with other agents and in this way, for example, share and/or retrieve information.

Multi Agent Systems (MAS) is a paradigm aligned with the bio-inspiration theories, comprising a community of agents with intelligence and behaviour emerging from the interaction between them. Agent-based solutions are suitable approaches to address the new requirements of flexibility, re-configurability and modularity. According to [Castle and Crooks, 2006] an agent-based approach also have advantages over traditional techniques, such as centralized and top-down approaches, related with caption of the emergent phenomena, from the flexibility and ability found in natural environments.

As stated before, in such distributed bio-inspired system, the behaviour of each entity follows simple few rules, but the overall emergent behaviour is very complex to understand and to demonstrate. Therefore, the design, test and simulation of distributed agent-based approaches, and particularly those exhibiting self-organizing and self-adaptive properties, are usually a hard task.

The use of computational platforms that simplifies these tasks and ensures a framework to simulate/validate strategies during the design phase assumes a crucial issue. For this purpose, Agent Based Modelling (ABM) tools provide an environment for programming, modelling and simulating agent-based solutions, aiming to test and compare alternative model configurations (e.g. alternative rules for individual behaviours) by reproducing a variety of patterns observed in the real system. The idea is to verify the correctness of the agent-based model at the design phase, correcting the identified mistakes and misunderstandings before its deployment into practical operation, using a set of richness scenarios. The use of such tools is very useful in the context of being less time consuming than having to write a program from scratch [Tobias and Hofmann, 2004].

4.1 Review and Evaluation of Agent Based Modelling Tools

A set of modelling and simulation environments are currently available for the simulation/validation of agent-based models exhibiting complex behaviour, namely MASON (http://cs.gmu.edu/~eclab/projects/mason/), Swarm (http://www.swarm.org/), NetLogo (http://ccl.northwestern.edu/netlogo/) and Repast (http://repast.sourceforge.net/). Several surveys reviewing and comparing well-known and widely used agent-based modelling and simulation platforms are available in the literature (see for example [Railsback et al., 2006] and [Allan, 2009]).

The goal in this work is not to evaluate each one of the existent ABM, but instead to make an overview based on studies already conducted and in the most popular platforms. Due to the open source policy of almost of the above ABMs (with the exception of the NetLogo) a more steady development is expected, having in consideration that all the community could make a contribution.

Agent development frameworks (e.g. JADE or JACK) aim to provide a platform to simplify the implementation of multi-agent systems, supporting also the debugging and deployment of the developed agent-based solutions. The main goal of this kind of tools is the development and deployment to real environment of agent-based systems, which differs from the primary objective ABM tools that aims to model and simulate agent-based system behaviour.

Mathematical systems (e.g. Matlab) can also be used to simulate multi-agent systems. These systems have an enormous drawback that is the need to build agent and their behaviour from the scratch. One big advantage is the possibility to use the almost endless mathematical potentialities of these tools. Other advantage, namely in Matlab, is the ability to use the simulated/validated code and deploy it to hardware.

4.1.1 Repast

Repast (Recursive Porous Agent Simulation Toolkit) was firstly developed at University of Chicago but is now maintained by the Argonne National Laboratory and managed by the Repast Organization for Architecture and Development (ROAD). Repast had 3 languages of implementation: Python (RepastPy), Java (RepastJ) and .NET (Repast.net): These 3 approaches have reached maturity and are not being developed but are still maintained [Castle and Crooks, 2006]. Now all these have been

superseded by the Repast Symphony (RepastS). Figure 6 gives an insight of RepastS integration with Eclipse IDE.

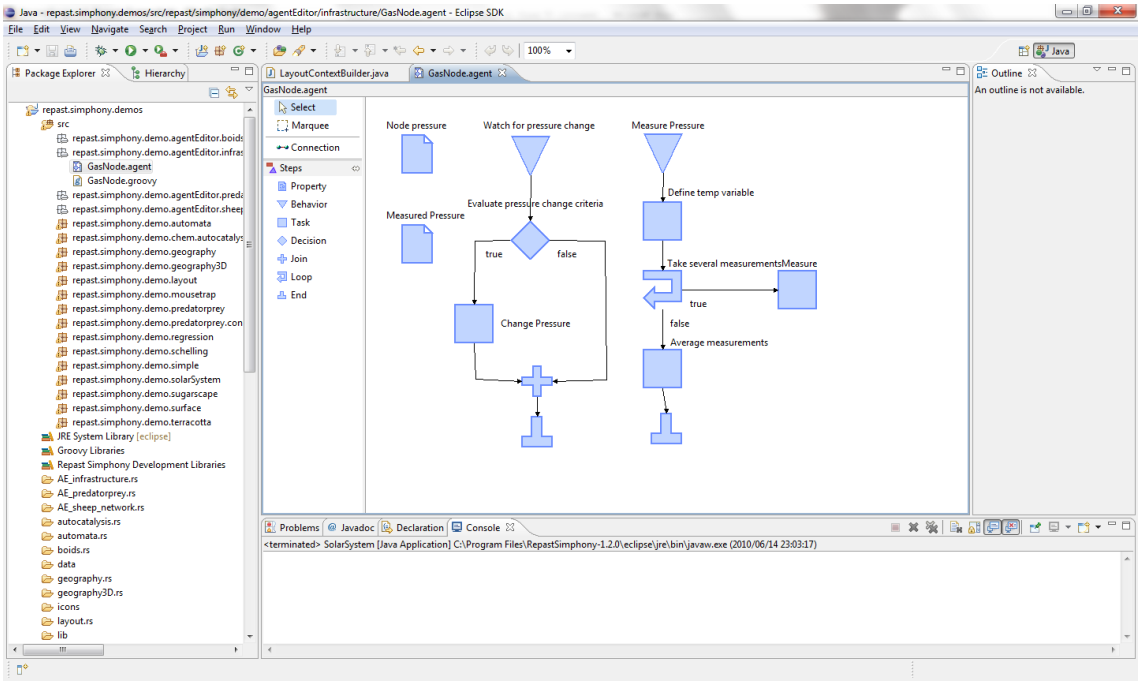

Figure 6 - RepastS with Eclipse IDE

RepastS has the advantage of being integrated into a well know IDE like Eclipse. It has a visual model development, visual model execution, automated database connectivity, automated output logging and results visualization [Allan, 2009].

Other advantage of RepastS is the possibility of integration with the mathematical tool Matlab.

4.1.2 MASON (Multi Agent Simulation Of Neighborhood)

MASON is being developed by the George Mason University's Computer Science Department and the George Mason University Center for Social Complexity. MASON is maybe the ABM with less maturity. MASON has been developed as an alternative to Repast and has the main goals of fast simulations and a large number of agents over a large number of iterations [Luke et al., 2005] and the reproducibility across hardware [Allan, 2009].

Regardless that being developed to maximize simulation speed and be an alternative to Repast it still is slower than this one in some simulations [Railsback et al., 2006]. Other disadvantage is the lack of a good GUI and also has the drawbacks of good

literature and a small group of users. One big advantage of MASON is the ability to stop simulations, copy data and restart the simulation in other machine [Allan, 2009]. Figure 7 shows a screenshot of an ant food foraging simulation running on MASON (as part of the package downloaded from [MASON, 2010]).

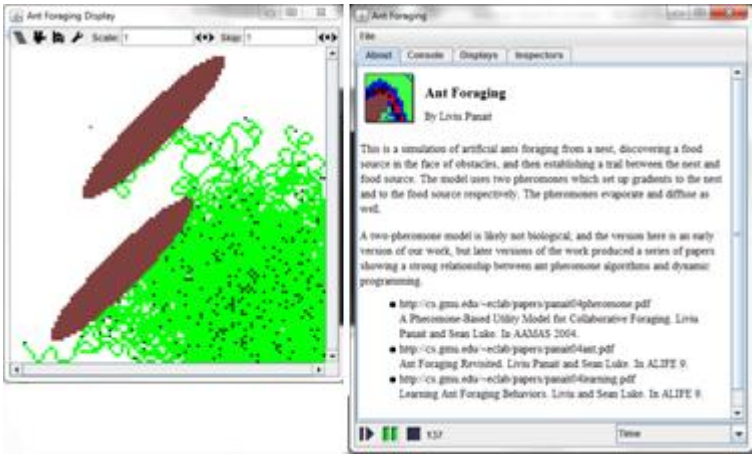

Figure 7 - MASON simulation example

Also according to [Berryman, 2008], MASON should be used when simulation speed and/or sophisticated batch runs are a required.

4.1.3 NetLogo

NetLogo comes from the "Network Logo" and has heritage the values of Logo programming language, developed in the 1960s by Seymour Papert. The assumption behind Logo is the ease of use (remember that the main audience for Logo were the children), but that doesn't mean that NetLogo is to be put apart in the world of ABM. In fact, because of the low knowledge of programming required, NetLogo is an excellent tool for starters and for the academic users.

The fantastic documentation, the examples available in the NetLogo library and the good users group feedback are also positive points. On favour is also the possibility of functionality extension through the use of API.

Some downsides of NetLogo are the simulation speed to a great number of agents and for code organization the fact that the code must be in a single file. Also while debugging it lacks a stepwise debugger.

The following figure is a screenshot of the behaviour of ant food foraging in the NetLogo environment (as part of the Models Library from [Wilensky, 1999])

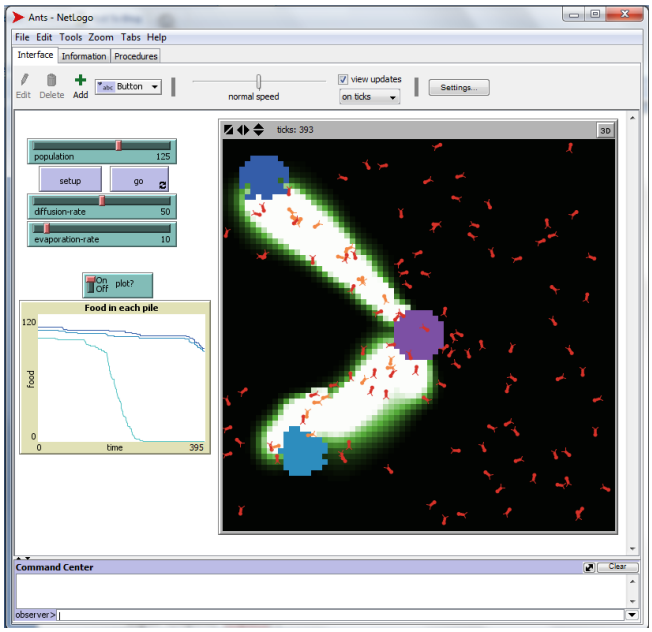

Figure 8 - NetLogo simulation example

According to [Railsback et al., 2006] NetLogo is a good choice if the model to build meets the paradigm of short-term, local interactions in a restricted space and not to extremely complex or if the model is to be implemented in lower-level platforms, because of the ease to use and fast programming.

4.1.4 Swarm

The Swarm platform was first developed in 1994 by Chris Langton at the Santa Fe Institute. Nowadays, is still being developed by a non-profit organization called Swarm development Group also based in Santa Fe.

Being an early starter, Swarm was build before the Java language be a reference. Therefore, the need of a programming that was less type consuming, like the one made in C++, was an issue and the developers opted by the Objective-C programming language. Meanwhile, a Java library was developed to simplify the use of Swarm and the next release will support Java Script end Scheme [Allan, 2009]. Figure 9 is a screenshot of an Artificial Stock Market model.

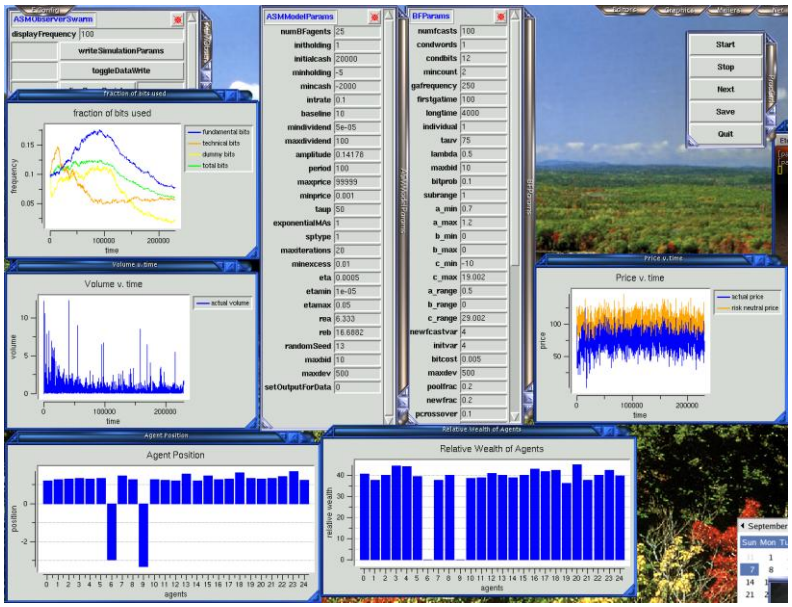

Figure 9 - Swarm screenshot (source: [Johnson, 2010])

According to [Allan, 2009], Swarm is still the most powerful and flexible simulation platform, but has a drawback that is the need of previous knowledge of Java or Objective-C programming language.

4.1.5 ABM comparison

Since currently several ABMs are available, a good choice has to be made taking into account several parameters, like the objective of the simulation, programming skills or the Graphical User Interface (GUI) facilities. The next table summarizes the studied ABMs according to some criteria [Barbosa and Leitão, 2010].

Table 3 - ABMs comparison

Name	***Mason***	***NetLogo***	***Swarm***	***Repast***
Availability (free)	yes	yes	yes	yes
Maturity	-	O	+	O
Programming effort	-	+	0	-
Change of properties	-	O	-	+
User interface	-	+	-	+
Simulation speed	+	O	O	+
Documentation	+	+	O	O

Legend: + Good; O Fair; - Poor

Analyzing the data in the Table 3, it is possible to conclude that the majority of existing platforms presents general weaknesses in terms of maturity, lack of support documentation and of statistical tools, and they will benefit from the integration with an integrated development environment (IDE), such as Eclipse (note that RepastS is already embedded in the Eclipse IDE). It is also possible to conclude that none of them is perfect, i.e. each one has good and weak points. For example, NetLogo is very good for the newcomers for its ease of programming but has some deficiencies like for example the simulation speed and practical limitations with the number of agents being executed. Some benefits of RepastS regarding the Swarm and MASON approach is the possibility of graphical construction of the model or parts of it. Regarding the GUI, Netlogo has a good, native, GUI and RepastS is built in Eclipse. Swarm and MASON being frameworks don't have a GUI.

These platforms are being used to simulate agent-based models for different application domains, such as economics, chemical, social behaviour and logistics. An interesting example in the manufacturing domain, described in [Sallez et al., 2009], is the use of the NetLogo platform to simulate the dynamic determination of the best path to route the products in situations characterized by the occurrence of disturbances.

In conclusion the choice of the correct ABM depends of the task to be performed and the skills of the person who will make that task. Since the experience of such programming skill is not very strong and taking into account the requirements specified, the NetLogo is more than well suited to be used in this work and therefore is the chosen tool. Also, the good documentation available make the NetLogo tool a good choice.

4.2 The NetLogo Modelling and Simulation Environment

Since the NetLogo tool was chosen to be used in this work, in this section a more detailed description of this tool is performed.

The NetLogo application runs on a Java Virtual Machine, therefore it is able to run on major platforms available (Windows, Linux, Mac, Solaris, etc). However, its programming language is based on the Logo programming language [Feurzeig et al., 1970], and not in Java, making it very easy to be used even by persons with low skills in programming.

NetLogo world is, basically, composed by two types of agents, the stationary agents (or patches) and the mobile agents (or turtles). The patches are arranged in a grid way, so they can form the world in over that the turtles move around. There is a third kind of agent that is the Link agent, which connects turtles so they can form networks, graphs and aggregates. NetLogo is fully customizable, for example, the user can set the size of the patches and/or the world. Another example of the type of customization is the ability to set the size, shape or colour of the turtles.

The GUI of NetLogo, see Figure 10, is structured in a tab way and is composed by 3 tabs: Interface, Information and Procedures. The *Interface tab* is the graphical part of NetLogo, i.e. it is in this part that the user can insert buttons, create graphics and see the world behaviour. The *Information tab* can be used to retrieve and/or change some information about the objective, functioning or bugs that the model may have. This is useful for the users (that are not the designer/developer) as a starting point to know the expected behaviour of the model. The *Procedures tab* is where the code is built, i.e. the creating the model with the desired characteristics and expected behaviour.

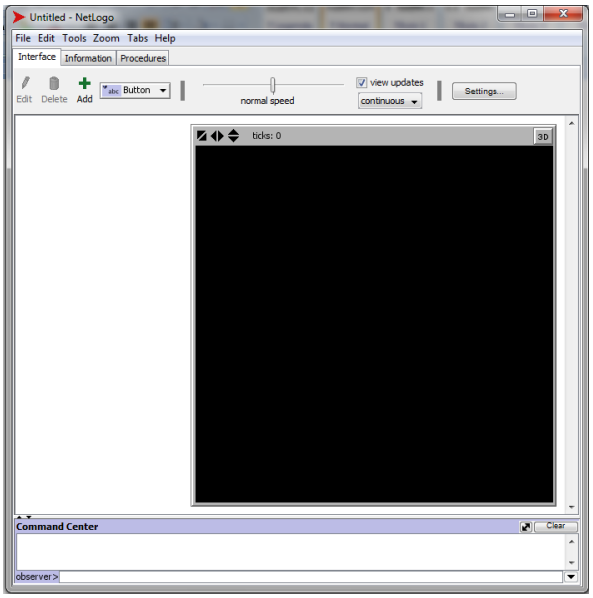

Figure 10 – NetLogo User Interface environment

The GUI has also a Command Center that permits the user to insert, in a live mode, a list of desired commands. This could be useful, for example to see if a certain variable has the correct value.

The graphic update rate could be made in two different ways: continuous mode and tick based mode. On continuous mode, the graphics are updated on continuous mode,

e.g., the user sees the world evolving when something changes. The disadvantage of this mode is the increase of the simulation time, due to the constant change of the world. If the model doesn't require a constant view update, the tick mode should be chosen. In this mode the view is only updated when the tick command appear in the Procedure tab. The tick mode also makes the simulation time processor independent, making the total returned time in ticks that is the same on every processor, for the same simulation.

NetLogo world, like the real world has North, South, East and West coordinates, has its own coordinates for the movement of the turtles. These coordinates will be important later on and are represented in the Figure 11 . As an example, if the desired movement of a turtle is on the upper way, the turtle must be oriented by 0º.

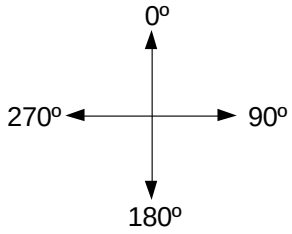

Figure 11 - NetLogo world coordinates

As stated before, is in the procedures tab that the programming is made. Just to give a few examples of the ease of programming see the following examples:

Table 4 – NetLogo encoding examples

Desired action	Encoding	Comments
Create an agent	`crt 1`	Which means "Create 1 turtle"
Move agent one patch upper way	`set heading 0` `fd 1`	Faces agent in upper way Move agent 1 patch
Check if patch ahead is empty	`if not any? product-on patch-ahead 1 []`	Checks if on the next patch is any agent called product
Remove first item from an array (e.g. service-list)	`set service-list remove-item 0 service-list`	Removes the first (0) item from the array named "service-list"
Count the total number of pallet on the system	`count product-on patches`	Counts the products (i.e. pallets) that are in the system (i.e. patches in NetLogo terminology)

For a more detailed description on NetLogo refer to [Wilensky, 1999].

5. Self-organizing Agent-based Model for an Automation System

During the development of this work an application to demonstrate the benefits of bio-inspired techniques for the industrial automation domain was developed. For that propose and based on the conclusions of the previous chapter, the NetLogo tool was selected.

5.1 Description of the Case Study

The case study corresponds to the FlexLink® Dynamic Assembly System (DAS) 30, depictured in Figure 12, which is a modular factory concept platform for assembly, inspection, test, repairing and packing applications [Leitão et al., 2010].

Figure 12 - FlexLink DAS 30 system (located at Schneider Electric GmbH in Seligenstadt, Germany)

The used DAS 30 transfer system layout is composed by several conveyors arranged in a closed-loop configuration. The main part of the transfer system is made of nine conveyors (C_1-C_9). These conveyors can be of the unidirectional and cross types. The unidirectional conveyor provides an input and an output port, and the cross conveyor provides transfers not only in the longitudinal but also in transversal axis. The system also has two lower conveyors (C_{10}-C_{11}). These two conveyors have the same behaviour as the normal unidirectional conveyors, but are physically longer to accomplish the distance (note that in the upper level there are three conveyors side to side). Two lifter units (L_1-L_2) are responsible for the interface between the upper and lower part of the system, and for transferring pallets into and out of the production system. Also notice that each lifter only can carry one palette at each time.

Figure 13 represents the system, contributing for a better understanding of its functioning, namely the possible directions at each conveyor.

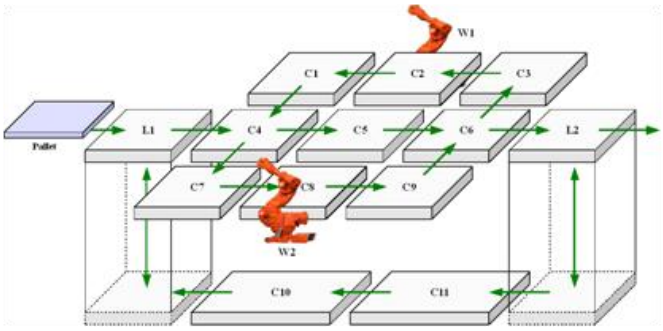

Figure 13 - Modular composition of the assembly system

The pallets move into the system through L1 and conveyor C4. From that conveyor the pallets move throughout the system with the possible alternatives. The conveyor C2 and C8 are associated to the two workstations, W1 and W2 respectively. The conveyors have the possibility to halt the pallets for processing. If the pallets have to leave the system they must go to conveyor C6 and, if L2 available, they exit.

The identification of pallets is done by using RFID (Radio-Frequency IDentification) technology. For this purpose, the conveyors C_2, C_4, C_6 and C_8 are equipped with RFID readers that are able to read/write information from/to RFID tags attached to the pallets.

When circulating in the system, a pallet is faced with several decision points, e.g. at C_4 and C_6. Considering a pallet at C_4, the decision point represents two alternative paths to convey the pallet, upon which it can either continue straight on, or turn in the direction of the workstation W_2. The decision to be taken is related to the service list to be performed and by the best path available.

5.2 Implementation of the Agent-based Model using NetLogo

The agent-based model designed within NetLogo for the described case study, which user interface is illustrated in Figure 14, comprises two types of agents: the pallets (e.g. turtles) and the resources (e.g. patches). Each one of these types of agents possesses a specific set of attributes and behaviors.

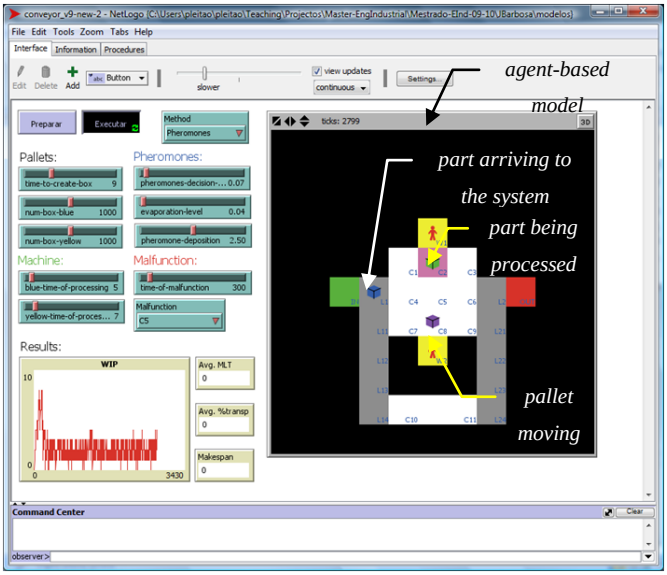

Figure 14 - NetLogo interface for the agent-based model

5.2.1 The Agents Attributes

The resources agents represent the conveyors, lifters and workstations included in the system. The resource agents representing conveyors have the following attributes: directions, pheromones_blue, pheromones_yellow, pheromones_W1, pheromones_W2, exit, velocity, availability and malfunction.

The *directions* attribute is a list of the possible ways that the conveyor to convey the pallet. Since in NetLogo the movements are in 2D the only possible moments are Up, Down, Left and Right (refer to Figure 11 for orientation). In this way, the list has the following format [0º 90º 180º 270º]. The possible movements have the respective number of orientation and the not allowed movements are signalled by putting "-1" in the correspondent position. As an example, the configuration for the conveyor C_4 is [-1 90 180 -1] which means that the conveyor can move the pallet forward or down.

The four attributes with respect to the pheromones are only used in the stigmergy mode (refer to chapter 5.2.2.3). They all look the same and all stores information about the best know path to a given objective. The difference between them lies precisely in the objective, e.g., pheromones_W1 gives the level of pheromone to get to the workstation 1 and the pheromones_blue, stores the level to the exit by the blue pallet.

The *exit* attribute indicates the exit of the system which means that all operations were made on a given pallet and it is out of the system and the *velocity* attribute gives the amount of time (in ticks) that a given conveyor takes to move one pallet. The *availability* and *malfunction* attributes gives the state of a given conveyor; in case of

negative in either of them, the conveyor cannot transport a pallet, being the first one related to the presence of a pallet and the second one to indicate the conveyor malfunction.

The pallet agent has the following main attributes: memory, birth time, death time, service list, orders, status, and time-of-processing.

The *memory* attribute stores the path, in the stigmergy mode, made by the pallet throughout the system, making this way possible to update later on the pheromone level in the right conveyors. The *birth time* and *death time* are attributes that stores the time of creation and exit time, respectively. These attributes are used to calculate some performance parameters.

The *service list* attribute contains the schedule of tasks for the pallet, as an example, it can have the following value [W1 OUT], meaning that the pallet must first go to the Workstation 1 and then exit the system. The *orders* attribute is used in the T-invariant mode, and stores the selected path (extracted from the T-invariant, as described later) to achieve the next objective in the service list. The *status* attribute acts like a flag that tells the pallet that is being processed by a workstation. The last attribute, *time-of-processing*, tells the workstation the amount of time that the pallet must be processed.

5.2.2 Implementation of the Agents Behaviour

These agents are regulated by a set of simple behavioural rules. As example, Figure 15 illustrates a flowchart that governs the behaviour of the pallet agents during their life-cycle. Basically, the pallet agents make movements within the system to execute the list of services required to perform the product that the pallet carries on. Before to make a movement, a pallet agent checks that the conveyor ahead is available (e.g. doesn't have any pallet) and is not breakdown. If the next movement involves one of the lifters it is also checked the above presumptions. The movement is only made if these two criteria are met. In case of more than one alternative path to evolve, the pallet agent applies a decision method to determine the best solution.

Additionally, it is possible to verify that if the goal is the execution of an operation in a workstation (i.e. that process the pallet), the pallet stays there until the processing is completed.

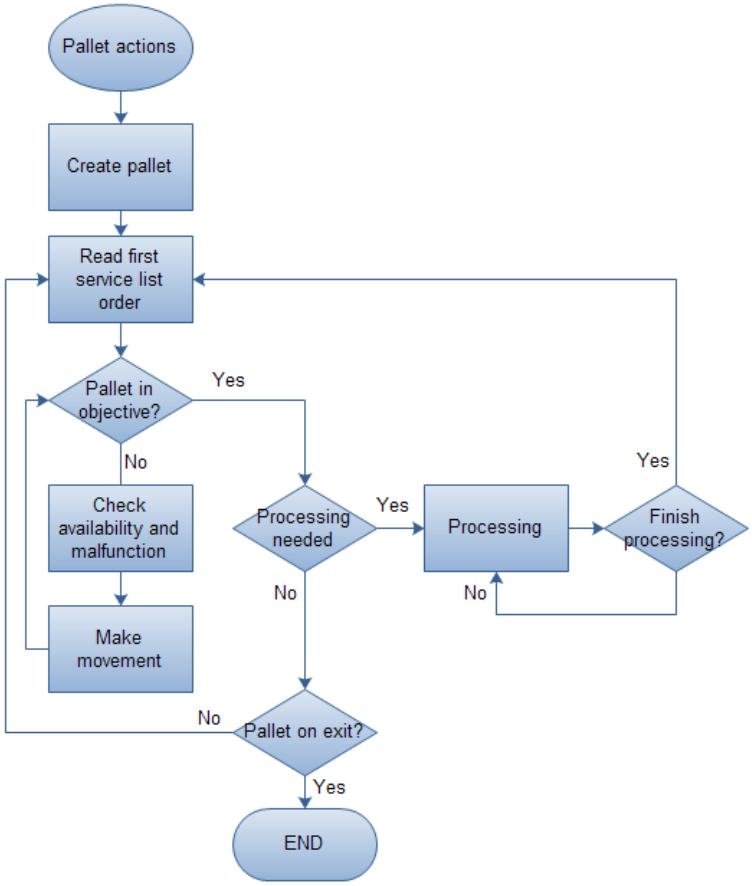

Figure 15 – Pallet movement execution flowchart

Each pallet agent has embedded functions to decide, in conflict situations, about the best path to be taken. In this work, three distinct functions were implemented [Barbosa and Leitão, 2010]: random, T-invariant and stigmergy modes.

5.2.2.1 Random mode

In the random mode, the pallet takes a simple random movement when facing more than one possible route to take. The random choice is made only with the available routes. For example, if the pallet is on the conveyor C_4 the random movement is taken considering only the 90º and 180 º movements. If the pallet is on the conveyor of the associated workstation and was not already machined, the pallet enters on processing mode, removing the first item of the service list.

When the service list is completed and the pallet is on the correspondent exit patch, the agents calculates some performance parameters (refer to chapter 5.2.5).

The implementation of the random mode is made of the following manner.

```
Procedure random
    read possible directions of present patch
    choose direction randomly
    make movement if patch available and not malfunctioning
End
```

In this pseudo code is only represented the functionality of the movements. For example, the processing part is not represented.

5.2.2.2 T-invariant mode

The second decision function is based on the work presented in [Leitão et al., 2010] and uses the knowledge extracted from the Petri nets models used to represent the system's layout, namely the T-invariants, x_i, that represent work cycles [Zurawski and Zhou, 1994]. The decision algorithm comprises two steps:

- *Identify the alternative paths*, by determining for each T-invariant, if it contains simultaneously the current location of the pallet and its destination location.
- *Choose the best path*, by evaluating the identified alternative paths taking into consideration a set of weighted criteria, namely distance, time and energy consumption.

As an example, considering a pallet located in C4 and aiming to go to the workstation located at C2, two different paths are available based on the following T-invariants:

- x_1={C_4, C_5, C_6, C_3, C_2, C_1}
- x_2={C_4, C_7, C_8, C_9, C_6, C_3, C_2, C_1}

These two paths will be evaluated according to a set of criteria, and the one that better fulfils the criteria will be selected. Note that these two paths are the elementary ones; others can be achieved considering the use of the lifters and the lower level of the system.

The implementation of the T-invariant mode is made of the following manner.

```
Procedure t-invariant
    If empty orders
          read first service list order
          search for path with start and finish patch
          store orders with t-invariant option
          remove first order from service list
    End if
```

```
        make movement if patch available and not malfunctioned
    End
```

In this pseudo code is only represented the functionality of the movements. For example, the processing part is not represented.

5.2.2.3 Stigmergy mode

The stigmergy decision mode is based on the concept of stigmergy. This mode consists in the lay down of signs by the pallets. The pallets when moving in the system memorize the path taken. After reach a goal (e.g. one workstation), the pallet update the correspondent pheromone level that lead to the goal. Before to update, the repeated moves are deleted, preventing the update of closed loops. Then, other pallets can then sense the most updated pheromone level and choose the best path.

In this mode the first pallets, due to the empty values of pheromones, move in a random manner, updating the trail every time a goal is reached.

Several parameters are used in the implementation of the pheromone mechanism, namely:

- *Pheromone deposition*, which reflects the intensity to be deposited in the patch (reinforcing the previous existing value). This value is weighted according to the path length, conducting to the selection of the shortest paths. As an example, if one given pallet takes more time than other to complete the same route, the pheromone deposition of the first one is less intense. In each patch there are four different levels of pheromones (as seen before). Each one respecting the correspondent objective.
- *Evaporation level*, which represents the natural process of evaporation along the time. This guarantees that the worst paths are dissipated if they are not reinforced by the other pallets.
- *Decision level*, this parameter represents the value, above which the pallet follows the pheromone trail and stops moving in a random manner.

If a pallet faces a situation of more than two alternative paths (e.g. with two pheromone levels above the decision level threshold) the pallet takes the path with a higher intensity of pheromone.

Regardless of the movement taken the pallet always stores the make movement.

The pheromones method can be encoded by the following pseudo code.

```
Procedure pheromones
   If empty orders
        read first service list order
        remove first order from service list
   End if
   read pheromone level to goal
   If pheromone level to objective  < pheromone decision level
        walk random if patch available and not malfunctioned
        store movement in memory
   Else
        find heading of maximum pheromone level
        make movement if patch available and not malfunctioned
        store movement in memory
   End if
End
```

In this pseudo code is only represented the functionality of the movements. For example, the processing part is not represented.

5.2.3 Implementation of the Graphical Aspects

The graphical aspect of the developed application is presented in Figure 16. In the left side are presented the user controls and the results area. The *users controls* are useful to set the parameters and simulation control. In the right pane is the depicted system where the evolution of the simulation can be observed.

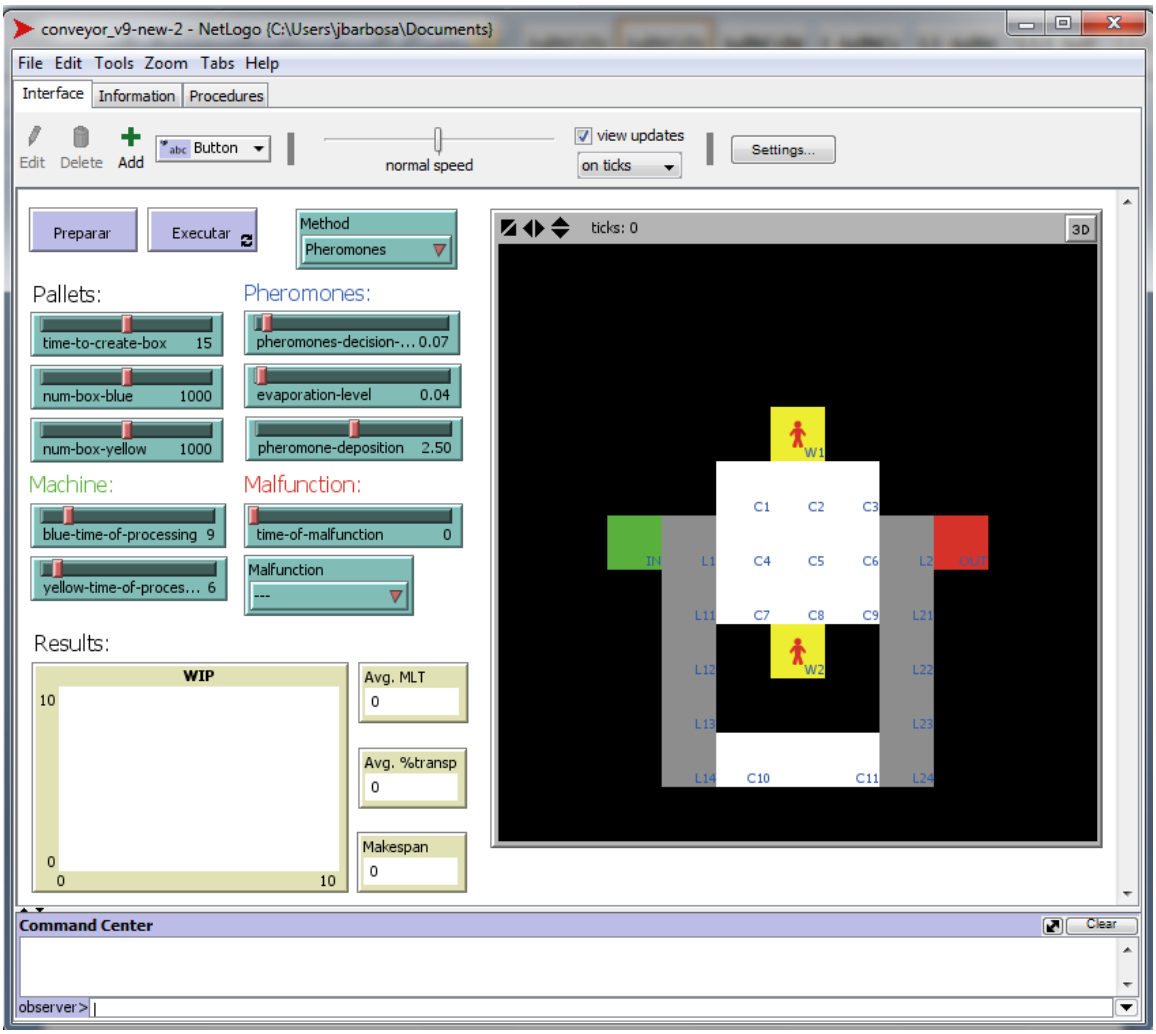

Figure 16 - GUI of the developed application

The decision method can be selected by the user through a *chooser button* (in the NetLogo nomenclature). This is done by selecting the correct decision mode before to start the simulation (see Figure 17). If desired, the method can also be changed in the middle of the simulation.

Figure 17 - Method chooser

In Figure 18 the parameters of the pallets are configured. The *time-to-create-box* parameter defines the time interval between the arrivals of pallets to the system. The creation time is given in ticks, meaning, with the example of the figure, that at each 15 ticks a pallet is created. Since we have two types of pallets, i.e. Blue and Yellow, the creation of the pallet is randomly, which means that could be a Blue or a Yellow one.

The other two parameters, i.e. *num-box-blue* and *num-box-yellow*, are used to set the number of each type of pallets to create.

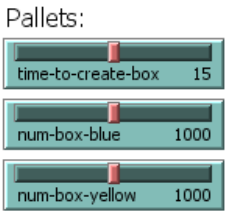

Figure 18 - Pallet creation parameters

The processing times of each pallet in the system resources are set using the sliders of Figure 19. In the given example the Blue pallets are processed during 9 ticks and the Yellow one by 6 ticks.

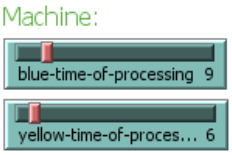

Figure 19 - Machine parameters

The malfunction zone gives the user the possibility to introduce a disturbance in the system at a given time during the simulation. For that propose, a time of malfunction must be set in conjunction with the malfunction conveyor. For example, if a malfunction is to be introduced at 1000 ticks at C_5, these parameters must be set at the slider and the chooser. If no malfunction is desired the *time-to-malfunction* parameter must be set to zero.

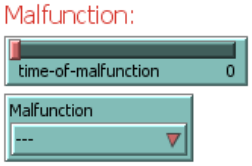

Figure 20 - Malfunction settings

The stigmergy decision mode uses some parameters that must be adjusted. As stated before, there are three parameters to be taken into consideration: pheromone-decision-level, evaporation-level and pheromone-deposition (for more information about the use of these parameters, please refer to 5.2.2.3). These parameters are configured using the sliders in the following figure.

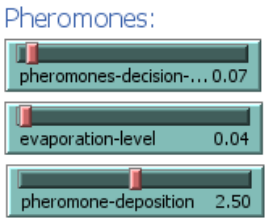

Figure 21 - Pheromones parameter adjustments

On the right side of the application interface (Figure 16), it is possible to visualize the behaviour of the system, namely the movement of the pallets, conveyors and lifters (see Figure 22).

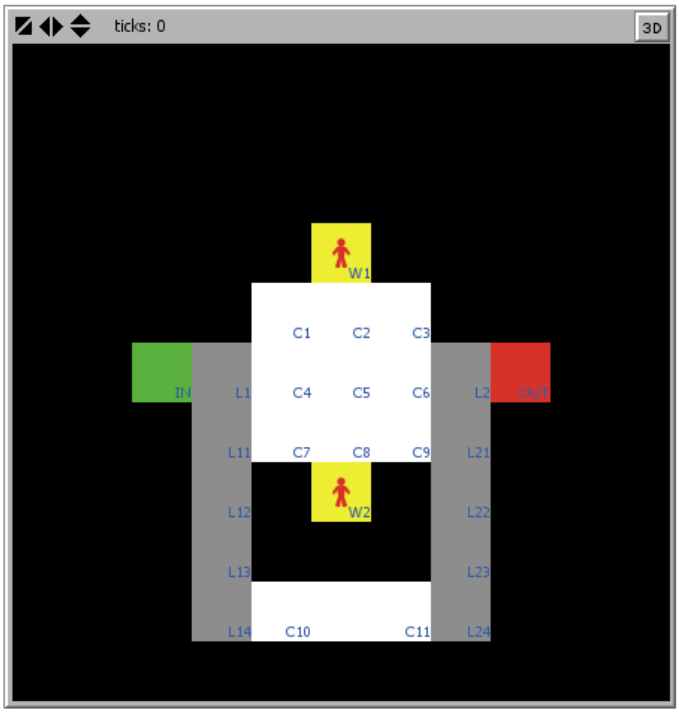

Figure 22 – The view of the system modeled in NetLogo

When the simulation is on run mode, the pallets movement can be observed and the expected behaviour confirmed.

5.2.4 Simulation Setup, Running and Global Behaviour

The simulation parameters are configured by pressing the *Setup* button (see Figure 23). When pressed, this button, configures, among other things, the size of the world, defines the conveyors behaviour and resets all values.

Figure 23 - Setup and Run buttons

In a pseudo code form, the setup procedure can be described as follows.

```
Procedure setup
    clear all
    setup graphical world
    setup conveyors parameters
    setup lifters parameters
    setup input and output
End
```

The *Run* button starts the simulation entering in a cycle which is only terminated when all created pallets have exited the system. Each cycle is started by checking the malfunctions, checking the machines that are currently processing and, if possible, generating a pallet. After that and depending of the chosen decision method, the correspondent procedure is called and a movement is made upon the decision rules. When all the pallets present in the system made their movement, the simulation time (i.e. tick count) is incremented and plots are updated. After all the pallets have exited the system, the variables are calculated and data is saved in a file.

In the same manner the run button executes the following pseudo code.

```
Procedure run
    While number of pallets  < total
          check for malfunctions
          check for machine processing
          generate random box
          If method = random
                call random procedure
          If method = t-invariant
                call t-invariant procedure
          If method = pheromones
                call pheromones procedure
          update tick cont
          update plots
    End while
    calculate variables
    save data to file
End procedure
```

5.2.5 Implementation of the Statistical Procedures

In all the previous methods, when the exit patch is reached by a pallet, some variables are calculated or updated. The total time since the pallet entered the system until it exits is calculated and stored in a global variable. Also, the total time subtracted by the manufacturing time is calculated and stored in a global variable.

If the simulation was running on the pheromone mode the level of pheromone in the travelled path is also updated. Before the update, the repeated moves are deleted, preventing the update of closed loops.

In a pseudo code form, the procedure that is executed after a pallet exit the system can be described as follows.

```
Procedure exit
    For each pallet
          calculate MLT
          calculate transport time
          If pheromone mode
                remove duplicated movements
                update pheromone level in patch
          End if
    End for
End
```

When the simulation is completed the average values of these variables are calculated and presented to the user. The first one is identified as average MLT and mean the time a pallet need to exit the system since their entrance. The second one is the average percentage transport time that gives the amount of time spent in transporting operations regarding the total time. Other parameters are the makespan that is the total time to execute all pallets and the Work in Process (WIP) that represents the amount of pallets that are simultaneously in the system. Here it is calculated the maximum value of the WIP and the average value.

These performance parameters are displayed in the Results area (see Figure 24).

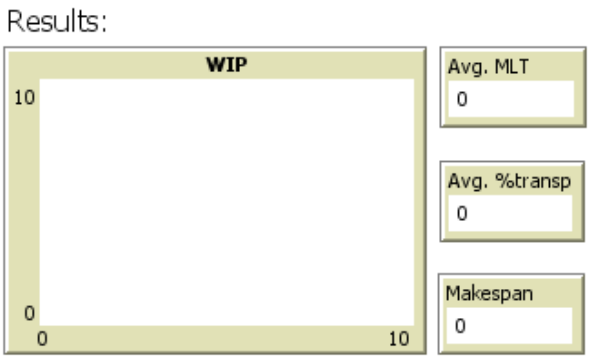

Figure 24 - Results area

All the values are returned at the end of the simulation execution, except the instantaneous value of the WIP that is plotted in the graphic along the time execution.

6. Analysis of the Experimental Results

The agent-based production control system modelled using the NetLogo is, at this stage, ready to be simulated under different scenarios, allowing evaluating different control strategies and alternative rules for the individual behaviour of the distributed agents. For this purpose, several experimental tests were performed to observe several performance parameters, such as the WIP, the MLP, the percentage of transport time for each pallet and the makespan.

The test scenario in the pallet concern is the following: the system is populated by pallets of two types, each one requiring the execution of one operation in one workstation. One pallet in the workstation W_1 (located at C_2) and the other one requiring the execution of one operation in the workstation W_2 (located at C_8). After the pallet is machined the pallet must follow to the exit of the system. The conveyors spend 1 time unit (t.u) to transfer the pallets, the lifters spend 5 t.u.to move the pallets and the machines W_1 and W_2 spends respectively 7 t.u. and 5 t.u. to execute their operations. In the pheromone mode, the configuration parameters are: 0,06 for the level of the pheromone decision, 0,03 for the evaporation level and 1,5 for the pheromone deposition. At this stage these parameters were tuned in an empirical manner after the observation of the system behaviour; in fact, changing these parameters implies a change in the system behaviour, e.g., if the pheromone decision level is too high, the pallets could move randomly indefinitely not considering the pheromone trails but if it is too low, the pallets could not sense the existence of pheromone trails.

During the first set of experimentations, the agent-based model was simulated for several arrival rates, without the occurrence of disturbances in the system and considering the different decision functions. From the experimental tests it was observable that the arrival rate, combined with the processing and transportation times of the mechatronic devices, influence the traffic jam in the production system. This is, if the arrival rate is too high, the system gets full and pallets have no available conveyors to go to. The performed simulations allowed the determination of the best value to flow the pallets along a production line. Note that being this experimental production system part of a production line, it is important to balance the production rate in order to avoid the existence of intermediate buffers (with pallets waiting to be processed) or the existence of machines waiting for pallets to be processed. The observed minimum time interval for the arrival of pallets at the input conveyor that guarantees the non-existence

of deadlocks or buffers in the system is 8 t.u., 9 t.u. and 15 t.u. for respectively, the stigmergy, T-invariant and random decision-making approaches.

Table 5a illustrates the experimental results for a scenario where the pallets arrive to the assembly system according to a deterministic function characterized by 16 t.u. of interval and without any disturbance occurrence.

Table 5 - Experimental results with breakdown at 300 t.u.

a) without disturbances

	Random	T-invariant	Stigmergy
avg (MLT)	54,54	14,00	17,35
avg (%Trans)	89,00	57,14	65,36
avg (WIP)	3,39	0,87	1,08
max (WIP)	8,70	2,00	3,20
makespan	32160	32061	32078

b) with disturbances (300 t.u.)

	Random	T-invariant	Stigmergy
avg (MLT)	59,53	17,61	19,29
avg (%Trans)	89,92	65,96	68,85
avg (WIP)	3,67	1,10	1,20
max (WIP)	9,70	2,00	3,70
makespan	32441	32049	32059

c) percentage of loss

	Random	T-invariant	Stigmergy
avg (MLT)	9,15%	25,79%	11,18%
avg (%Trans)	1,04%	15,43%	5,35%
avg (WIP)	8,19%	26,44%	11,08%
max (WIP)	11,49%	0,00%	15,63%
makespan	0,87%	-0,04%	-0,06%

The differences between the T-invariant and the stigmergy approaches with the random mode can be better seen in Figure 25 (which is a graphical representation of Table 5a). Here can be observed that the results obtained in the random mode are worst than the other two approaches, with the last ones presenting very close performance results.

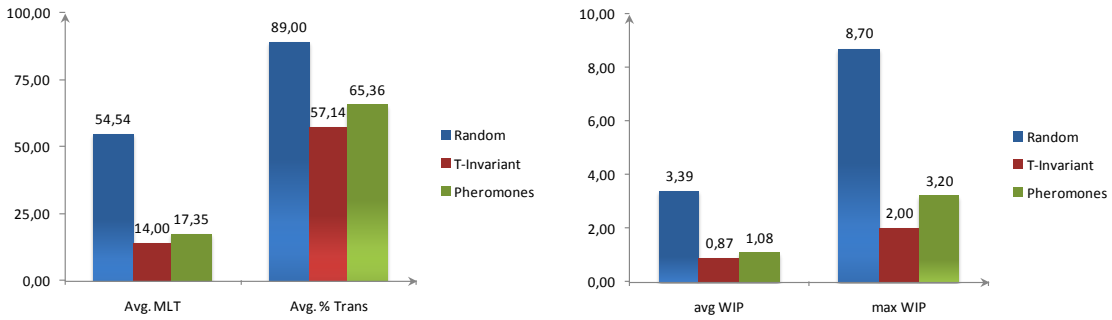

Figure 25 – Graphical results without breakdown

A second set of experimentations was performed to evaluate the response of the agent-based control model in presence of disturbances. Table 5**Erro! A origem da referência não foi encontrada.**b illustrates a scenario that considers the same assumptions of the previous one but now a breakdown on the conveyor C5 will occur, which implies that pallets should choose alternative paths to reach their goals (C_5 became unavailable at time 300 t.u.).

The results obtained in this experimental test show that the presence of disturbances strongly affects the WIP and MLT parameters in the several approaches. As illustrated in Table 5c it is possible to verify that the stigmergy method is the one that presents less loss of performance in presence of disturbances, reflecting its inherent capabilities to dynamically self-adapt to changing environments. In fact, each pallet using the stigmergy method tries to find alternative paths to execute its objective dynamically and on the fly, minimizing the effects of the disturbance.

Figure 26 shows in a graphic manner the obtained performed results. It is also possible to observe that, with breakdown at 300 t.u., the results between the best two approaches are even more close than the previous ones (without malfunction) and the difference to the random approach is still considerable.

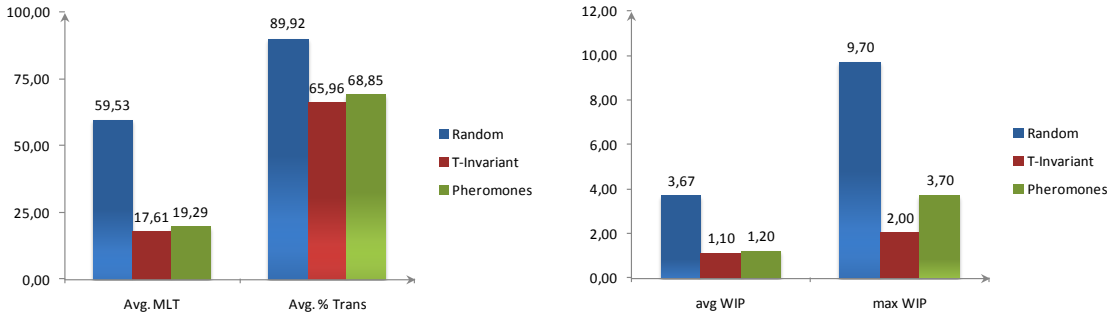

Figure 26 - Graphical results with breakdown at 300 t.u.

Again, the stigmergy method is the most robust approach since it presents the best value for the minimum interval for the arrival of pallets, namely 8 t.u., 9 t.u. and 16 t.u. for respectively the stigmergy, T-invariant and random decision-making methods.

A third batch of simulations was made with the presence of disturbances but in this case, the conveyor C5 becomes unavailable at half time of the makespan of each mode without disturbances. As can be observed in Table 6b, the mode based on the T-invariant approach is the one that still presents the overall best results. After the analysis

of the Table 6c, which illustrates the percentage of loss, the stigmergy approach is the one that exhibits the less percentage of lost (except for the maximum WIP which can be explained by the fact of the readjustment of the pheromones levels to the new routes, i.e., during the readjustment period the pallets still don't have the best path performance and therefore the system is more occupied). Note that the conveyor breakdown occurs at a time when the paths are very well defined.

Table 6 – Experimental results with breakdown at half of the makespan

a) without disturbances

	Random	T-invariant	Stigmergy
avg (MLT)	54,54	14,00	17,35
avg (%Trans)	89,00	57,14	65,36
avg (WIP)	3,39	0,87	1,08
max (WIP)	8,70	2,00	3,20
makespan	32160	32061	32078

b) with disturbances (50% of t.u.)

	Random	T-invariant	Stigmergy
avg (MLT)	57,33	15,84	18,79
avg (%Trans)	89,53	62,08	68,05
avg (WIP)	3,55	0,99	1,17
max (WIP)	9,30	2,00	4,40
makespan	32284	32045	32052

c) percentage of loss

	Random	T-invariant	Stigmergy
avg (MLT)	5,12%	13,14%	8,30%
avg (%Trans)	0,60%	8,64%	4,12%
avg (WIP)	4,69%	13,56%	8,40%
max (WIP)	6,90%	0,00%	37,50%
makespan	0,39%	-0,05%	-0,08%

Figure 27 also shows that, with breakdown at half of the makespan without breakdown, the results between the best two approaches are still very close and the difference to the random approach is still considerable.

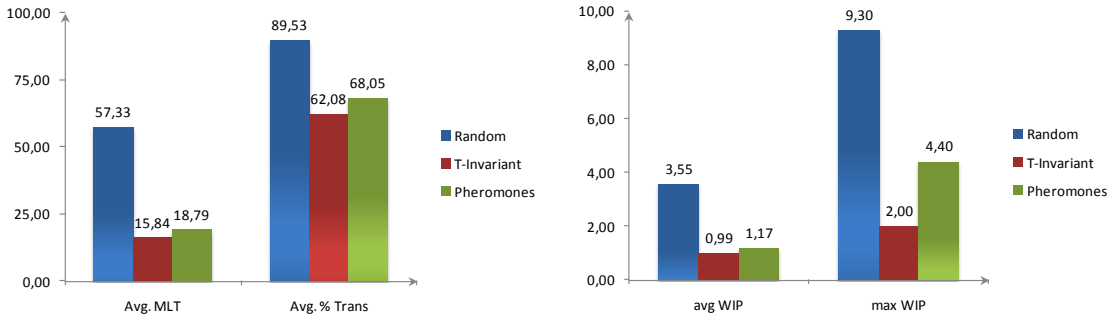

Figure 27 - Graphical results with breakdown at half of makespan

In all cases, the decision function based on the T-invariant mode presents, as expected, better optimization, since it was performed using a pre-defined global view, but it presents some lacks in terms of re-configurability of the system layout. In case of need to change or reconfigure the system layout it is necessary to update the Petri nets model and consequently to extract a new set of T-invariants. This approach can address the static, off-line re-configuration but not the on-line reconfiguration (dynamically performed at run time) [Leitão et al., 2010].

Since, one important feature to be reached is the dynamic system re-configurability, the stigmergic approach shows to be a promising approach, combining the good performance with flexibility, adaptability and robustness. In fact, this approach seems able to cope with changing environments, in particular the product variability and the layout reconfiguration, due to its decentralized nature and adaptability capability.

It is also very important to compare the evolution of the WIP for the three decision approaches in order to try to get some complementary conclusions about the system performance (see Figure 28).

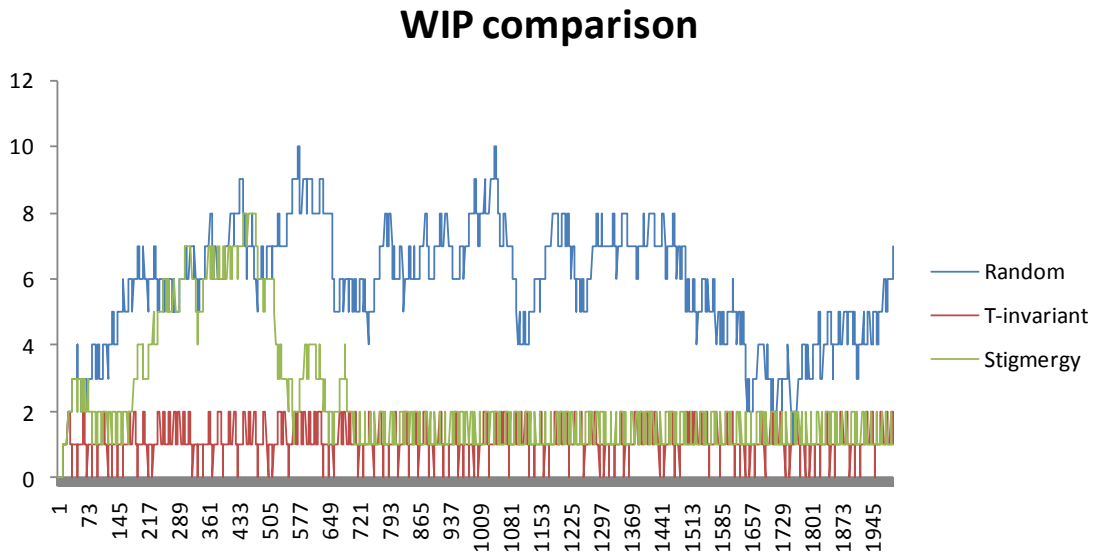

Figure 28 – Graphical evolution of WIP

The observation of the graphic tells that the T-invariant approach is the one that guarantees a value of the WIP most constant and the Random approach is the one that have a most inconstant value of the WIP, due to its random nature. In beginning of the Stigmergy mode it is possible to verify a climbing of the WIP. This is a consequence of the learning phase, i.e., in this period the pallets are still moving in a random way and

the pheromones levels are below the decision threshold. The WIP becomes constant when the pheromone levels are being considered to the pallet movement.

The experimental implementation allowed concluding that the use of a programming and modelling environment to simulate agent-based control solutions in general and for manufacturing and automation in particular, contributes for a fast prototyping and proof of concept [Barbosa and Leitão, 2010]. In fact, the designed agent-based model can be easily simulated under different scenarios (some of them impossible or difficult to be created in real world) to validate/improve/optimize control strategies before its practical realization.

7. Conclusions

The work developed in this project aimed to study bio-inspired solutions to solve several complex problems from different domains and in particular in the manufacturing and automation domains.

In this work some of the existing bio-inspired methodologies to solve complex mathematical/engineering problems are analyzed and discussed. Among them are analysed the ants based algorithms, such as ACO and PSO, the bees algorithms, GA algorithms, self-organization and artificial immune systems. The conclusion of this study is that these mechanisms are a good inspiration to be mimicked to solve numerous daily problems. It is of good practice for the researchers to try to see the part of the behaviour of interest to be mimicked and not always mimic the whole behaviour if is not necessary.

Some of the investigation and real life applications of bio-inspired algorithms are enumerated. The analysis of the constant growth on the number of publications made in these areas allows concluding that this is an area of enormous growing capacity and of great interest. Also, promising results are obtained by the majority of the investigations, making bio-inspired solutions close or even better than the known solutions to a give problem.

A brief review of some of the most popular ABMs tools was made. It was concluded that these tools are of great interest to evaluate and/or to be a proof of concept to bio-inspired algorithms, making this last ones better understood. From the existing ABM platforms, NetLogo was selected to develop an agent-based application for a production system, which uses a stigmergy based approach to implement the decision algorithm to route pallets within the conveyor system.

The several simulations made on the implemented agent-based application proved that even the simplest implementation of an algorithm based on the behaviour of ants can achieve good results compared to the optimal, T-invariant, solution. It was shown that this simple implementation is much better than the random solution. Also, if a more elaborated algorithm were implemented the expected results would have the tendency to be better and closer to the best know solution.

As future work, the parameters of the stigmergy approach should be more extensively tested and refined in order to obtain even better results. An optimization for

batch simulations should be developed to easily collect data of a great number of simulations rounds. Also, the modelling and simulation of more complex scenarios should be performed, e.g., the behaviour of an identical system coupled to the exit of the previous one.

More bio-inspired algorithms should be implemented and conclusions drawn on the best one to this kind of engineering problem. Another important issue to be further researched in the future is the integration of the modelled agent-based systems with the real physical world.

Bibliography

[Abd-El-Barr et al., 2003] Abd-El-Barr, M., S. Sait, and B. Sarif (2003). Ant colony algorithm for evolutionary design of arithmetic circuits. *Proceedings of the 15th International Conference on Microelectronics, 2003. ICM 2003.*, 198 – 201.

[Aggoune et al., 2001] Aggoune, R., A. Mahdi, and M.-C. Portmann (2001). Genetic algorithms for the flow shop scheduling problem with availability constraints. *Proceedgins of the IEEE International Conference on Systems, Man, and Cybernetics, 2001 4*, 2546 –2551 vol.4.

[Ai and Kachitvichyanukul, 2009] Ai, T. and V. Kachitvichyanukul (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research 36*, 1693–1702.

[Albert et al., 2009] Albert, F., S. Koh, C. Chen, C. Loo, and S. Tiong (2009). Path control of dexterous robotic hand using genetic algorithm. *Proceedings of the 4th International Conference on Autonomous Robots and Agents, 2009. ICARA 2009.*, 502 –506.

[Allan, 2009] Allan, R. J. (2009). Survey of agent based modelling and simulation tools.

[Arnaout et al., 2008] Arnaout, J.-P., R. Musa, and G. Rabadi (2008). Ant colony optimization algorithm to parallel machine scheduling problem with setups. *Proceeding of the IEEE International Conference on Automation Science and Engineering, 2008. CASE 2008*, 578 –582.

[Aziz et al., 2007] Aziz, N., A. Mohemmed, and B. Daya Sagar (2007). Particle swarm optimization and voronoi diagram for wireless sensor networks coverage optimization. *Proceeding of the International Conference on Intelligent and Advanced Systems, 2007. ICIAS 2007*, 961 –965.

[Badawy et al., 2005] Badawy, F., H. Abdelazim, and M. Darwish (2005). Genetic algorithms for predicting the egyptian stock market. *Proceeding ot the Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on Information and Communications Technology, 2005*, 109 –122.

[Bae et al., 2001] Bae, J. I., D. C. Lee, D. S. Ahn, J. M. Lee, K. E. Kim, and M. S. Kim (2001). Speed control of fork lift vehicle using a genetic algorithm.

Proceedings of the IEEE International Symposium on Industrial Electronics, 2001. Proceedings. ISIE 2001. 3, 1839 –1844 vol.3.

[Barbosa and Leitão, 2010] Barbosa, J. and P. Leitão (2010). Modelling and simulating self-organizing agent-based manufacturing systems. *to appear in the Proceedings of the 36th Annual Conference of the IEEE Industrial Electronics Society (IECON'10) Arizona, US, 8-10 November*.

[Beckers et al., 1992] Beckers, R., J. Deneubourg, and S. Goss (1992). Trails and u-turns in the selection of the shortest path by the ant lasius niger. *Journal of theoretical biology 159*, 397–415.

[Bell and McMullen, 2004] Bell, J. E. and P. R. McMullen (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics 18*, 41–48.

[Berryman, 2008] Berryman, M. J. (2008). Review of software platforms for agent based models. Technical report, Defence Science and Technology Organisation, Edinburgh, Australia.

[Blum and Sampels, 2004] Blum, C. and M. Sampels (2004). An ant colony optimization algorithm for shop scheduling problems. *Journal of Mathematical Modelling and Algorithms 3*, 285–308.

[Bonabeau et al., 1999] Bonabeau, E., M. Dorigo, and G. Theraulaz (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford.

[Boubertakh et al., 2009] Boubertakh, H., M. Tadjine, P.-Y. Glorennec, and S. Labiod (2009). Tuning fuzzy pid controllers using ant colony optimization. *Proceeding of the 17th Mediterranean Conference on Control and Automation, 2009. MED '09.*, 13 –18.

[Bousbia and D. Trentesaux, 2002] Bousbia, S. and D. D. Trentesaux (2002). Self-organization in distributed manufacturing control: State-of-the-art and future trends. *Proceeding of the IEEE International Conference on Systems, Man and Cybernetics 5*.

[Brown and McShane, 2004] Brown, J. and M. McShane (2004). Optimal design of nanoengineered implantable optical sensors using a genetic algorithm.

Proceedings of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2004. IEMBS '04. 1, 2105 –2108.

[Bussmann and Sieverding, 2001] Bussmann, S. and J. Sieverding (2001). Holonic control of an engine assembly plant - an industrial evaluation. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 169–174.

[Caldeira et al., 2007] Caldeira, J., R. Azevedo, C. Silva, and J. Sousa (2007). Supply-chain management using aco and beam-aco algorithms. *Proceedings of the IEEE International Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007.*, 1 –6.

[Camazine et al., 2001] Camazine, S., J. L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau (2001). *Self-Organization in Biological Systems*. Princeton University Press.

[Camilo et al., 2006] Camilo, T., C. Carreto, J. S. Silva, and F. Boavida (2006). An energy-efficient ant-based routing algorithm for wireless sensor networks. *ANTS Workshop*, 49–59.

[Castle and Crooks, 2006] Castle, C. J. E. and A. T. Crooks (2006). Principles and concepts of agent-based modelling for developing geospatial simulations, centre for advanced spatial analysis (university college london): Working paper 110.

[Castro and Timmis, 2002] Castro, L. and J. Timmis (2002). *Artificial Immune Systems: A New Computational Intelligence Approach*. London: Springer-Verlag.

[Chandramouli and Izquierdo, 2006] Chandramouli, K. and E. Izquierdo (2006). Image classification using chaotic particle swarm optimization. *Proceedings of the IEEE International Conference on Image Processing, 2006*, 3001 –3004.

[Chen et al., 2009] Chen, A.-P., C.-H. Huang, and Y.-C. Hsu (2009). A novel modified particle swarm optimization for forecasting financial time series. *Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. 1*, 683 –687.

[Chen and Rogers, 2009] Chen, G. and K. Rogers (2009). Proposition of two multiple criteria models applied to dynamic multi-objective facility layout problem based on ant colony optimization. *Proceedgins of the IEEE International Conference on Industrial Engineering and Engineering Management, 2009. IEEM 2009.*, 1553 –1557.

[Chen et al., 2008] Chen, R.-M., S.-T. Lo, C.-L. Wu, and T.-H. Lin (2008). An effective ant colony optimization-based algorithm for flow shop scheduling. *Proceedgins of the IEEE Conference onSoft Computing in Industrial Applications, 2008. SMCia '08.*, 101 –106.

[Chen et al., 2009] Chen, Y.-W., A. Mimori, and C.-L. Lin (2009). Hybrid particle swarm optimization for 3-d image registration. *Proceedings of the 16th IEEE International Conference on Image Processing (ICIP), 2009*, 1753 –1756.

[Cheng et al., 2009] Cheng, C.-T., K. Fallahi, H. Leung, and C. Tse (2009). Cooperative path planner for uavs using aco algorithm with gaussian distribution functions. *Proceedings of the IEEE International Symposium on Circuits and Systems, 2009. ISCAS 2009.*, 173 –176.

[Colson et al., 2009] Colson, C., M. Nehrir, and C. Wang (2009). Ant colony optimization for microgrid multi-objective power management. *Proceedings of the IEEE/PES Power Systems Conference and Exposition, 2009. PSCE '09.*, 1 –7.

[Corry and Kozan, 2004] Corry, P. and E. Kozan (2004). Ant colony optimisation for machine layout problems. *Computational Optimization and Applications 28*, 287–310.

[Cui and Potok, 2007] Cui, X. and T. Potok (2007). A particle swarm social model for multi-agent based insurgency warfare simulation. *Proceedings of the 5th ACIS International Conference on Software Engineering Research, Management Applications, 2007. SERA 2007.*, 177 –183.

[Das and Bhattacharya, 2009] Das, A. and M. Bhattacharya (2009). A study on prognosis of brain tumors using fuzzy logic and genetic algorithm based techniques. *Proceedgins of the International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing, 2009. IJCBS '09.*, 348 –351.

[Deneubourg et al., 1990] Deneubourg, S. Aron, S. Goss, and J. M. Pasteels (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior 3*(2), 159–168.

[Di Caro and Dorigo, 1998] Di Caro, G. and M. Dorigo (1998). Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research 9*, 317–365.

[Dong et al., 2007] Dong, Q., S. Kan, L. Qin, and Z. Huang (2007). Sequencing mixed model assembly lines based on a modified particle swarm optimization multi-objective algorithm. *Proceedings of the IEEE International Conference on Automation and Logistics, 2007*, 2818 –2823.

[Dongming et al., 2008] Dongming, Z., X. Kewen, W. Baozhu, and G. Jinyong (2008). An approach to mobile ip routing based on qpso algorithm. *Proceedings of the Pacific-Asia Workshop on Computational Intelligence and Industrial Application, 2008. PACIIA '08. 1*, 667 –671.

[Dorigo, 1992] Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms (in Italian)*. Ph. D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

[Dorigo, 2007] Dorigo, M. (2007). Ant colony optimization.

[Dorigo and Stützle, 2009] Dorigo, M. and T. Stützle (2009). Ant colony optimization: Overview and recent advances. Technical Report TR/IRIDIA/2009-013, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

[Duan et al., 2009] Duan, H., X. Zhang, J. Wu, and G. Ma (2009). Max-min adaptive ant colony optimization approach to multi-uavs coordinated trajectory replanning in dynamic and uncertain environments. *Journal of Bionic Engineering 6*, 161–173.

[Eberhart and Kennedy, 1995] Eberhart, R. C. and J. Kennedy (1995). A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, 39–43.

[Elmahi et al., 2004] Elmahi, I., S. Merzouk, O. Grunder, and A. Elmoudni (2004). A genetic algorithm approach for the batches delivery optimization in a supply chain. *Proceedings of the IEEE International Conference on Networking, Sensing and Control, 2004 1*, 299 – 304 Vol.1.

[Fang and Bai, 2009] Fang, X. and T. Bai (2009). Share price prediction using wavelet transform and ant colony algorithm for parameters optimization. *SVM, WRI Global Congress on Intelligent Systems 3*, 288–292.

[Farmer et al., 1986] Farmer, J., N. Packard, and A. Perelson (1986). The immune system, adaptation and machine learning. *Physica D 2*, 187—204.

[Fausett, 1994] Fausett, L. (1994). *Fundamentals of neural networks: architectures, algorithms, and applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

[Feurzeig et al., 1970] Feurzeig, W., S. Papert, M. Bloom, R. Grant, and C. Solomon (1970). Programming-languages as a conceptual framework for teaching mathematics. *SIGCUE Outlook 4*(2), 13–17.

[Frisch, 1967] Frisch, K. v. (1967). *The Dance Language and Orientation of Bees*. Cambridge, Mass.: The Belknap Press of Harvard University Press.

[Gaing, 2004] Gaing, Z. L. (2004). A particle swarm optimization approach for optimum design of pid controller in avr system. *Proceedings of the IEEE Transactions on Energy Conversion 19, no.2*, 384–391.

[Gao and Chen, 2008] Gao, N. and W. D. Chen (2008). A genetic algorithm for disassembly scheduling with assembly product structure. *Proceedings of the IEEE International Conference on Service Operations and Logistics, and Informatics, 2008. IEEE/SOLI 2008. 2*, 2238 –2243.

[Goss et al., 1989] Goss, S., S. Aron, J. Deneubourg, and J. Pasteels (1989). Self-organized shortcuts in the argentine ant. *Naturwissenschaften 76*(12), 579–581.

[Goudos et al., 2008] Goudos, S., I. Rekanos, and J. Sahalos (2008). Emi reduction and ics optimal arrangement inside high-speed networking equipment using particle swarm optimization. *Proceedings of the IEEE Transactions on Electromagnetic Compatibility 50*(3), 586 –596.

[Grassé., 1959] Grassé., P. P. (1959). La reconstruction du nid et les coordinations inter-individuelles chez bel-licositermes natalensis et cubitermes sp. la théorie de la stigmergie: Essai d'interpréta-tion du comportement des termites constructeurs. *Insectes Sociaux 6*, 41–84.

[Gravel et al., 2002] Gravel, M., W. Price, and C. Gagné (2002). Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *European Journal of Operational Research 143*, 218–229.

[Hadeli et al., 2004] Hadeli, P. Valckenaers, M. Kollingbaum, and H. Van Brussel (2004). Multi-agent coordination and control using stimergy. *Computers in Industry 53*, 75–96.

[Holland, 1999] Holland, J. H. (1999). *Emergence: From Chaos to Order*. Perseus Publishing.

[Hong, 2009] Hong, L. (2009). A novel artificial immune algorithm for job shop scheduling. *Proceedings of the International Conference on Computational Intelligence and Natural Computing, 2009. CINC '09. 1*, 38 –41.

[Hong et al., 2007] Hong, W.-C., Y.-F. Chen, P.-W. Chen, and Y.-H. Yeh (2007). Continuous ant colony optimization algorithms in a support vector regression based financial forecasting model. *Proceedings of the Third International Conference on Natural Computation, 2007. ICNC 2007. 1*, 548 –552.

[Hu et al., 2005] Hu, H., Q. Hu, Z. Lu, and D. Xu (2005). Optimal pid controller design in pmsm servo system via particle swarm optimization. *Proceedings of the 31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.*, 5 pp.

[Hu, 2010] Hu, X. (2010). PSO tutorial. http://www.swarmintelligence.org/tutorials.php.

[Jain and Sharma, 2005] Jain, P. and P. Sharma (2005). Solving job shop layout problem using ant colony optimization technique. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2005 1*, 288 – 292 Vol. 1.

[Jalilvand et al., 2008] Jalilvand, A., A. Kimiyaghalam, A. Ashouri, and M. Mahdavi (2008). Advanced particle swarm optimization-based pid controller parameters tuning. *Proceedings of the IEEE International Multitopic Conference, 2008. INMIC 2008.*, 429 –435.

[Jiang et al., 2009] Jiang, B., W. Zhang, and P. Zhang (2009). Research on an improved genetic algorithm which can improve the node positioning optimized solution of wireless sensor networks. *Proceedgins of the International Conference on Computational Science and Engineering, 2009. CSE '09. 2*, 949 –954.

[Jianhua and Xianfeng, 2010] Jianhua, W. and H. Xianfeng (2010). A hybrid genetic algorithm for agile supply chain scheduling optimization. *Proceedings of the 2nd International Conference on Future Computer and Communication (ICFCC), 2010 1*, V1–396 –V1–400.

[Johnson, 2010] Johnson, P. E. (2010, available at August 27th). Artificial stock market.

[Jun, 2009] Jun, L. (2009). A genetic algorithm to logistics distribution vehicle routing problem with fuzzy due time. *Proceedings of the Sixth International Conference on Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. 6*, 33 – 39.

[Kaijun et al., 2010] Kaijun, L., C. Nanfang, and W. Yuxia (2010). Genetic optimization of retailer selection in supply chain management. *Proceedings of the 2nd IEEE International Conference on Information Management and Engineering (ICIME), 2010*, 124 –127.

[Karlra and Prakash, 2003] Karlra, P. and N. Prakash (2003). A neuro-genetic algorithm approach for solving the inverse kinematics of robotic manipulators. *Proceedgins of the IEEE International Conference on Systems, Man and Cybernetics, 2003. 2*, 1979 – 1984 vol.2.

[Katayama et al., 2006] Katayama, K., K. Mackin, K. Matsushita, and E. Nunohiro (2006). Applying brightness information in satellite image data search using distributed genetic algorithm. *Proceedings of the International Conference on Hybrid Information Technology, 2006. ICHIT '06. 2*, 84 –89.

[Khanna et al., 2006] Khanna, R., H. Liu, and H.-H. Chen (2006). Self-organization of sensor networks using genetic algorithms. *Proceedings of the IEEE International Conference on Communications, 2006. ICC '06. 8*, 3377 –3382.

[Kulkarni and Shanker, 2007] Kulkarni, P. and K. Shanker (2007). A genetic algorithm for layout problems in cellular manufacturing systems. *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management, 2007*, 694 –698.

[Lazzerini et al., 1999] Lazzerini, B., F. Marcelloni, G. Dini, and F. Failli (1999). Assembly planning based on genetic algorithms. *Proceedings of the 18th*

International Conference of the North American Fuzzy Information Processing Society, 1999. NAFIPS., 482 –486.

[Lee et al., 1997] Lee, J.-H., Y. Choi, B.-T. Zhang, and C. S. Kim (1997). Using a genetic algorithm for communication link partitioning. *Proceedings of the IEEE International Conference on Evolutionary Computation, 1997.*, 581 –584.

[Lee and Vlachogiannis, 2005] Lee, K. and J. Vlachogiannis (2005). Optimization of power systems based on ant colony system algorithms: An overview. *Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems, 2005.*, 22 –35.

[Lee et al., 2009] Lee, Y., S.-C. Cheng, C.-C. Chang, and C.-H. Chuang (2009). An ant colony optimization algorithm for dna copy number analysis in array cgh data. *Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09.*, 842 –847.

[Leeton et al., 2010] Leeton, U., T. Ratniyomchai, and T. Kulworawanichpong (2010). Optimal reactive power flow with distributed generating plants in electric power distribution systems. *Proceedings of the International Conference on Advances in Energy Engineering (ICAEE), 2010*, 166 –169.

[Lei et al., 2003] Lei, J., Y. Yamada, and Y. Komura (2003). Layout optimization of manufacturing cells using particle swarm optimization. *Proceedings of the SICE 2003 Annual Conference 1*, 392 – 396 Vol.1.

[Leitao, 2008] Leitao, P. (2008). Self-organization in manufacturing systems: Challenges and opportunities. *Proceedings of th Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, 2008. SASOW 2008.*, 174 –179.

[Leitão, 2009a] Leitão, P. (2009a). Agent-based distributed manufacturing control: A state-of-the-art survey. *Proceedings of the International Journal of Engineering Applications of Artificial Intelligence 22, n. 7*, 979–991.

[Leitão, 2009b] Leitão, P. (2009b). Holonic rationale and self-organization on design of complex evolvable systems. *Proceedings of the 4th Int'l. Conf. on Industrial Applications of Holonic and Multi-Agent Systems, LNAI 5696, Springer*, 13–24.

[Leitão et al., 2010] Leitão, P., J. Alves, J. Mendes, and A. Colombo (2010). Energy aware knowledge extraction from petri nets supporting decision-making in service-oriented automation. *to appear in the Proceedings of the IEEE International Symposium on Industrial Electronics*.

[Leitão and Restivo, 2006] Leitão, P. and F. Restivo (2006). Adacor: A holonic architecture for agile and adaptive manufacturing control. *Computers in Industry 57 (2)*, 121–130.

[Li and Pi, 2009] Li, C. and Y. Pi (2009). A hybrid of particle swarm optimization and ensemble learning for credit risk assessment. *Proc. of the Int'l. Conf. on Computational Intelligence and Software Engineering*, 1–4.

[Li et al., 2008] Li, T., M. Yang, S. Chen, Z. Zhao, and Z. Ge (2008). On qos anycast routing algorithm based on particle swarm optimization. *Proceedings of the 9th International Conference for Young Computer Scientists, 2008. ICYCS 2008.*, 386 –391.

[Li and Lei, 2009] Li, Z. and L. Lei (2009). Sensor node deployment in wireless sensor networks based on improved particle swarm optimization. *Proceedings of the International Conference on Applied Superconductivity and Electromagnetic Devices, 2009. ASEMD 2009.*, 215 –217.

[Lima et al., 2007] Lima, M., A. Araujo, and A. Cesar (2007). Adaptive genetic algorithms for dynamic channel assignment in mobile cellular communication systems. *Proceedings of the IEEE Transactions on Vehicular Technology 56*(5), 2685 –2696.

[Liu and Ge, 2008] Liu, H. and S. Ge (2008). Reactive power optimization based on improved particle swarm optimization algorithm with boundary restriction. *Proceedings of the Third International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, 2008. DRPT 2008.*, 1365 –1370.

[Liu et al., 2005] Liu, H., Z. Xu, and A. Abraham (2005). Hybrid fuzzy-genetic algorithm approach for crew grouping. *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, 2005. ISDA '05.*, 332 – 337.

[Liu et al., 2009] Liu, Z., D. Zhao, X. Zhang, B. Dan, and F. Guan (2009). Reactive power optimization in power system based on chaos ant colony algorithm. *Proceedings of the International Conference on Sustainable Power Generation and Supply, 2009. SUPERGEN '09.*, 1 –4.

[Logeswari and Karnan, 2010] Logeswari, T. and M. Karnan (2010). An improved implementation of brain tumor detection using soft computing. *Proceedings of the Second International Conference on Communication Software and Networks, 2010. ICCSN '10.*, 147 –151.

[Lu et al., 2008] Lu, C., H. Huang, B. Zheng, J. Fuh, and Y. Wong (2008). An ant colony optimization approach to disassembly planning. *Proceedings of the International Conference on Apperceiving Computing and Intelligence Analysis, 2008. ICACIA 2008.*, 81 –84.

[Luke et al., 2005] Luke, S., C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan (2005). Mason: A multiagent simulation environment. *SIMULATION 81*(7), 517–527.

[Lv and Lu, 2009] Lv, H. and C. Lu (2009). A discrete particle swarm optimization algorithm for assembly sequence planning. *Proceedings of the 8th International Conference on Reliability, Maintainability and Safety, 2009. ICRMS 2009.*, 1119 –1122.

[Ma et al., 2008] Ma, M., Y. Zhang, H. Tian, and Y. Lu (2008). A fast sar image segmentation algorithm based on particle swarm optimization and grey entropy. *Proceedings of the Fourth International Conference on Natural Computation, 2008. ICNC '08. 4*, 8 –12.

[Majhi et al., 2008] Majhi, R., G. Panda, G. Sahoo, A. Panda, and A. Choubey (2008). Prediction of s&p 500 and djia stock indices using particle swarm optimization technique. *Proceedings of the IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence).*, 1276 – 1282.

[Malisia and Tizhoosh, 2006] Malisia, A. and H. Tizhoosh (2006). Image thresholding using ant colony optimization. *Proceedings of the 3rd Canadian Conference on Computer and Robot Vision*.

[MASON, 2010] MASON (2010). Multi agent simulation of neighborhood. Accessed May 25th, 2010.

[Massotte, 1995] Massotte, P. (1995). Self-organization: A new approach to improve the reactivity of the production systems. *Proc. of the IEEE International Conference on Emergent Technologies for Factory Automation*, 23–32.

[Matlock et al., 2009] Matlock, A., R. Holsapple, C. Schumacher, J. Hansen, and A. Girard (2009). Cooperative defensive surveillance using unmanned aerial vehicles. *Proceedings of the American Control Conference, 2009. ACC '09.*, 2612 –2617.

[Maulik, 2009] Maulik, U. (2009). Medical image segmentation using genetic algorithms. *Proceedings of the IEEE Transactions on Information Technology in Biomedicine 13*(2), 166 –173.

[Meng, 2006] Meng, Y. (2006). A swarm intelligence based algorithm for proteomic pattern detection of ovarian cancer. *Proceedgins of the IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB '06.*, 1 –7.

[Miller, 2007] Miller, P. (2007). The genius of swarms. *National Geographic*.

[Montana et al., 1999] Montana, D., G. Bidwell, G. Vidaver, and J. Herrero (1999). Scheduling and route selection for military land moves using genetic algorithms. *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99. 2*, 1123 Vol. 2.

[Moore and Sinclair, 1999] Moore, S. and M. Sinclair (1999). Design of routing tables for a survivable military communications network using genetic algorithms. *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99. 3*, 1795 Vol. 3.

[Mori et al., 1998] Mori, K., M. Tsukiyama, and T. Fukuda (1998). Adaptive scheduling system inspired by immune system. *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics, 1998. 4*, 3833 –3837 vol.4.

[Moudada et al., 2004] Moudada, F., M. Bonani, M. S., A. Guignard, and D. Floreano (2004). Physical connections and cooperation in swarm robotics. *Proceedings of the 8th Conf. on Intelligent Autonomous Systems*, 53–60.

[Márkus et al., 1996] Márkus, A., T. K. Váncza, and L. Monostori (1996). A market approach to holonic manufacturing. *CIRP Annals - Manufacturing Technology 45*(1), 433 – 436.

[Munirajan et al., 2004] Munirajan, V., F. Sahin, and E. Cole (2004). Ant colony optimization based swarms: implementation for the mine detection application. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2004 1*, 716 –721 vol.1.

[Muraleedharan and Osadciw, 2009] Muraleedharan, R. and L. Osadciw (2009). An intrusion detection framework for sensor networks using ant colony. *Proceedings of the Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers, 2009*, 275 –278.

[Ning et al., 2004] Ning, L., L. Fei, S. Debao, and H. Chang (2004). Particle swarm optimization for constrained layout optimization. *Proceedings of the Fifth World Congress on Intelligent Control and Automation, 2004. WCICA 2004. 3*, 2214 – 2218 Vol.3.

[Niu et al., 2007] Niu, X., Y. Qiu, and S. Tong (2007). Application of particle swarm system as a novel parameter optimization technique on spatiotemporal retina model. *Proc. of the 29th Annual Int'l. Conf. of the IEEE Engineering in Medicine and Biology Society*, 5794–5797.

[Nouyan et al., 2009] Nouyan, S., R. Gross, M. Bonani, F. Mondada, and M. Dorigo (2009). Teamwork in self-organized robot colonies. *Proceedings of the IEEE Transactions on Evolutionary Computation 13*, 695–711.

[Ohmori et al., 2010] Ohmori, S., K. Yoshimoto, and K. Ogawa (2010). Solving facility layout problem via particle swarm optimization. *Proceedings of the Third International Joint Conference on Computational Science and Optimization (CSO), 2010 1*, 409 –413.

[Parunak, 1996] Parunak, H. V. (1996). *Foundations of Distributed Artificial Intelligence*, Chapter Applications of Distributed Artificial Intelligence in Industry, pp. 139–164. John Wiley & Sons.

[Parunak, 2005] Parunak, H. V. D. (May 2005). Expert assessment of human-human stigmergy. Technical report, Analysis for the Canadian Defence Organization, Altarum Institute, Ann Arbor, Michigan.

[Peña et al., 2006] Peña, J., A. Upegui, and E. Sanchez (2006). Particle swarm optimization with discrete recombination: An online optimizer for evolvable hardware. *Proc. of the 1st NASA/ESA Conf. on Adaptive Hardware and Systems*, 163–170.

[Peng et al., 2000] Peng, H., F. Long, Z. Chi, and W. Su (2000). A hierarchical distributed genetic algorithm for image segmentation. *Proceedings of the 2000 Congress on Evolutionary Computation, 2000. 1*, 272 –276 vol.1.

[Pessin et al., 2009] Pessin, G., F. Osorio, D. Wolf, and M. Dias (2009). Genetic algorithm applied to robotic squad coordination. *Proceedings of the Electronics, Robotics and Automotive Mechanics Conference, 2009. CERMA '09.*, 169 –174.

[Pham et al., 2007] Pham, D., A. A., and E. Koc (2007). Manufacturing cell formation using the bees algorithm. *Proc. of the IPROMS Innovative Production Machines and Systems Virtual Conference*.

[Pham et al., 2007] Pham, D., E. Koc, J. Y. Lee, and J. Phrueksanant (2007). Using the bees algorithm to schedule jobs for a machine. *Proc. of the 8th Int'l. Conf. on Laser Metrology, CMM and Machine Tool Performance*, 430–439.

[Pham et al., 2005] Pham, D. T., G. A, K. E, O. S, R. S, and Z. M. (2005). The bees algorithm. Technical report, Manufacturing Engineering Centre, Cardiff University, UK.

[Poli, 2007] Poli, R. (May-November 2007). An analysis of publications on particle swarm optimization applications. Technical report, CSM-469, Department of Computing and Electronic Systems, University of Essex, Colchester, Essex, UK.

[Qi et al., 2008] Qi, E.-S., J.-Y. Chen, and L. Liu (2008). Supply chain partners selection based on rvpk algorithm. *Proceedings of the 4th International*

Conference on Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08., 1 –4.

[Qiu et al., 2009] Qiu, H., W. Zhou, and H. Wang (2009). A genetic algorithm-based approach to flexible job-shop scheduling problem. *Proceedings of the Fifth International Conference on Natural Computation, 2009. ICNC '09. 4*, 81 –85.

[Railsback et al., 2006] Railsback, S. F., S. L. Lytinen, and S. K. Jackson (2006). Agent-based simulation platforms: Review and development recommendations. *SIMULATION 82*(9), 609–623.

[Ramirez-Rosado and Bernal-Agustin, 1998] Ramirez-Rosado, I. and J. Bernal-Agustin (1998). Genetic algorithms applied to the design of large power distribution systems. *Proceedings of the IEEE Transactions on Power Systems 13*(2), 696 –703.

[Regue et al., 2001] Regue, J.-R., M. Ribo, J.-M. Garrell, and A. Martin (2001). A genetic algorithm based method for source identification and far-field radiated emissions prediction from near-field measurements for pcb characterization. *Proceedings of the IEEE Transactions on Electromagnetic Compatibility 43*(4), 520 –530.

[Ren and Cheng, 2009] Ren, B. and L. Cheng (2009). Smt automatic optical inspection path planning based on mdspso algorithm. *Proceedings of the International Conference on Computational Intelligence and Natural Computing, 2009. CINC '09. 2*, 134 –137.

[Sallez et al., 2009] Sallez, Y., T. Berger, and D. Trentesaux (2009). A stigmergic approach for dynamic routing of active products in fms. *Computers in Industry 60*, 204–216.

[Samanta and Nataraj, 2009] Samanta, B. and C. Nataraj (2009). Application of particle swarm optimization and proximal support vector machines for fault detection. *Swarm Intelligence 3, n. 4*, 303–325.

[Satheesh Kumar et al., 2009] Satheesh Kumar, R., P. Asokan, and S. Kumanan (2009). Artificial immune system-based algorithm for the unidirectional loop layout problem in a flexible manufacturing system. *The International Journal of Advanced Manufacturing Technology 40*, 553–565. 10.1007/s00170-008-1375-y.

[Sethuram and Parashar, 2006] Sethuram, R. and M. Parashar (2006). Ant colony optimization and its application to boolean satisfiability for digital vlsi circuits. *Proceedings of the International Conference on Advanced Computing and Communications, 2006. ADCOM 2006.*, 507 –512.

[Shan et al., 2007] Shan, H., S. Li, J. Huang, Z. Gao, and W. Li (2007). Ant colony optimization algorithm-based disassembly sequence planning. *Proceedings of the International Conference on Mechatronics and Automation, 2007. ICMA 2007.*, 867 –872.

[Sharma et al., 2009] Sharma, S., R. Mohapatra, B. Biswal, and B. Choudhury (2009). Generation of robotic assembly sequence using ant colony optimization. *Industrial and Information Systems (ICIIS), 2009 International Conference on*, 520 –525.

[Shi et al., 2009] Shi, D., J. He, and L. Wang (2009). Job shop scheduling problem with a novel particle swarm optimization based on tabu search. *Proceedgins of the International Conference on Artificial Intelligence and Computational Intelligence, 2009. AICI '09. 2*, 97 –100.

[Sim and Sun, 2002] Sim, K. M. and W. H. Sun (2002). Multiple ant-colony optimization for network routing. *Proceedgins of the First International Symposium on Cyber Worlds, 2002. Proceedings.*, 277 – 281.

[Sinha et al., 2009] Sinha, A., H. Aditya, M. Tiwari, and F. Chan (2009). Multi-agent based petroleum supply chain coordination: A co-evolutionary particle swarm optimization approach. *Proceedings of the World Congress on Nature Biologically Inspired Computing, 2009. NaBIC 2009.*, 1349 –1354.

[Skinner, 2010] Skinner, M. (2010). Genetic algorithms overview.

[Sun et al., 2008] Sun, R., X. Wang, and G. Zhao (2008). An ant colony optimization approach to multi-objective supply chain model. *Proceedings of the Second International Conference on Secure System Integration and Reliability Improvement, 2008. SSIRI '08.*, 193 –194.

[Sun and Teng, 2002] Sun, Z.-G. and H.-F. Teng (2002). An ant colony optimization based layout optimization algorithm. *Proceedings of the 2002 IEEE Region 10*

Conference on Computers, Communications, Control and Power Engineering TENCON '02. 1, 675 – 678 vol.1.

[Suva et al., 2004] Suva, C., I. Sousa, J. Sa da Costa, and T. Runkler (2004). A multi-agent approach for supply chain management using ant colony optimization. *Proceeddings of the IEEE International Conference on Systems, Man and Cybernetics, 2004 2*, 1938 – 1943 vol.2.

[Teodorovic, 2008] Teodorovic, D. (2008). Swarm intelligence systems for transportation engineering: Principles and applications. *Transportation Research - Part C 16*, 651–667.

[Tewolde et al., 2008] Tewolde, G., D. Hanna, and R. Haskell (2008). Hardware pso for sensor network applications. *Proceedings of the Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*, 1 –8.

[Thamarajah, 1998] Thamarajah, A. (1998). A self-organizing model for scheduling distributed autonomous manufacturing systems. *Cybernetics Systems 29 (5)*, 461–480.

[Thangaraj et al., 2009] Thangaraj, R., M. Pant, and A. Nagar (2009). Maximization of expected target damage value using quantum particle swarm optimization. *Proceedings of the Second International Conference on Developments in eSystems Engineering (DESE), 2009*, 329 –334.

[Tian et al., 2008] Tian, J., W. Yu, and S. Xie (2008). An ant colony optimization algorithm for image edge detection. *Proceedings of the IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence).*, 751 –756.

[Timmis et al., 2008] Timmis, J., A. Hone, T. Stibor, and E. Clark (2008). Theoretical advances in artificial immune systems. *Theoretical Computer Science 403*(1), 11 – 32.

[Tobias and Hofmann, 2004] Tobias, R. and C. Hofmann (2004). Evaluation of free java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation 7*.

[Toderici et al., 2010] Toderici, M., S. Toderici, and M. Imecs (2010). Optimization of induction motor control using genetic algorithms. *Proceedings of the 2010 IEEE*

International Conference on Automation Quality and Testing Robotics (AQTR), 3, 1 –6.

[Tohka et al., 2007] Tohka, J., E. Krestyannikov, I. Dinov, A. Graham, D. Shattuck, U. Ruotsalainen, and A. Toga (2007). Genetic algorithms for finite mixture model based voxel classification in neuroimaging. *Proceedings of the IEEE Transactions on Medical Imaging 26*(5), 696 –711.

[Tong et al., 2004] Tong, Z., L. Ning, and S. Debao (2004). Genetic algorithm for vehicle routing problem with time window with uncertain vehicle number. *Proceedings of the Fifth World Congress on Intelligent Control and Automation, 2004. WCICA 2004. 4*, 2846 – 2849 Vol.4.

[Tsai and Chou, 2006] Tsai, J.-T. and J.-H. Chou (2006). Circuit tolerance design using an improved genetic algorithm. *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision, 2006. ICARCV '06.*, 1 – 6.

[Tunjongsirigul and Pongchairerks, 2010] Tunjongsirigul, B. and P. Pongchairerks (2010). A genetic algorithm for a vehicle routing problem on a real application of bakery delivery. *Proceedings of the 2010 International Conference on Electronic Computer Technology (ICECT)*, 214 –217.

[Vaario and Ueda, 1996] Vaario, J. and K. Ueda (1996). Self-organisation in manufacturing systems. *Proceedings of the Japan- USA Symposium on Flexible Automation, Boston, US*, 1481–1484.

[Van Ast et al., 2009] Van Ast, J., R. Babuska, and B. De Schutter (2009). Fuzzy ant colony optimization for optimal control. *Proceedings of the American Control Conference, 2009. ACC '09.*, 1003 –1008.

[Wai and Su, 2006] Wai, R.-J. and K.-H. Su (2006). Supervisory control for linear piezoelectric ceramic motor drive using genetic algorithm. *Proceedings of the IEEE Transactions on Industrial Electronics 53*(2), 657 – 673.

[Wang et al., 2008] Wang, G., Y. Yan, X. Zhang, J. Shangguan, and Y. Xiao (2008). A simulation optimization approach for facility layout problem. *Proc. of the IEEE Int'l. Conf. on Industrial Engineering and Engineering Management*, 734–738.

[Wang et al., 2005] Wang, X.-N., Y.-J. Feng, and Z.-R. Feng (2005). Ant colony optimization for image segmentation. *Proceedings of the International Conference on Machine Learning and Cybernetics, 2005. 9*, 5355 –5360 Vol. 9.

[Wilensky, 1999] Wilensky, U. (1999). Netlogo. http://ccl.northwestern.edu/netlogo/.

[Wu and Tan, 2009] Wu, J. and Y. Tan (2009). A particle swarm optimization algorithm for grain logistics vehicle routing problem. *Proceedings of the ISECS International Colloquium on Computing, Communication, Control, and Management, 2009. CCCM 2009. 3*, 364 –367.

[Xia and Wu, 2005] Xia, W. and Z. Wu (2005). A hybrid particle swarm optimization approach for job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 360–366.

[Xu et al., 2009] Xu, D.-S., X.-Y. Ai, and L.-N. Xing (2009). An improved ant colony optimization for flexible job shop scheduling problems. *Proceedings of the International Joint Conference on Computational Sciences and Optimization, 2009. CSO 2009. 1*, 517 –519.

[Yamada et al., 2003] Yamada, Y., K. Ookoudo, and Y. Komura (2003). Layout optimization of manufacturing cells and allocation optimization of transport robots in reconfigurable manufacturing systems using particle swarm optimization. *Proc. of the Int'l. Conf. on Intelligent Robots and Systems*.

[Yuan and Zou, 2009] Yuan, X.-e. and Y. Zou (2009). Technology program financial forecast model based on caco-svm. *Proceedings of the International Workshop onIntelligent Systems and Applications, 2009. ISA 2009.*, 1 –4.

[Zhang et al., 2008] Zhang, B., Y. Yang, and L. Gan (2008). Dynamic control of wind/photovoltaic hybrid power systems based on an advanced particle swarm optimization. *Proceedings of the IEEE International Conference on Industrial Technology, 2008. ICIT 2008.*, 1 –6.

[Zhang et al., 2008] Zhang, J., H. S. Chung, A. W. Lo, and T. Huang (2008). Optimization of power electronic circuits using ant colony system. *Proceedings of the IEEE Transactions on Power Electronics 24*, 147–162.

[Zhang and hai Wang, 2008] Zhang, Q. and X. hai Wang (2008). Region water supply system optimization based on binary and continuous ant colony algorithms. *Proceedings of the International Conference on Intelligent Computation Technology and Automation (ICICTA), 2008 1*, 130 –134.

[Zhang and Wu, 2008] Zhang, R. and C. Wu (2008). An effective immune particle swarm optimization algorithm for scheduling job shops. *Proceedings of the 3rd IEEE Conference on Industrial Electronics and Applications, 2008. ICIEA 2008.*, 758 –763.

[Zhao et al., 2009] Zhao, D., L. Luo, and K. Zhang (2009). An improved ant colony optimization for communication network routing problem. *Proceedings of the Fourth International Conference on Bio-Inspired Computing, 2009. BIC-TA '09.*, 1 –4.

[Zhengxiong and Xinsheng, 2010] Zhengxiong, G. and G. Xinsheng (2010). A particle swarm optimization for the motion planning of wheeled mobile robot. *Proceedings of the 8th World Congress on Intelligent Control and Automation (WCICA), 2010*, 2410 –2414.

[Zurawski and Zhou, 1994] Zurawski, R. and M. Zhou (1994). Petri nets and industrial applications: A tutorial. *Proceedings of the IEEE Transactions on Industrial Electronics 41*(6), 567 –583.

Attachments

```
globals [
  tapetes
  lifter1
  lifter2
  list_length
  new_directions
  OUT-blue
  OUT-yellow
  count-box-blue
  count-box-yellow
  exit-found?
  exit-total
  possible-exits
  aux-directions
  product-exit-time
  average-product-exit-time
  distance-to-jump
  P1
  P2
  P3
  P4
  P5
  P6
  P7
  P8
  P9
  P10
  P11
  P12
  P13
  next-heading
  create-next-box?
  IN-free
  last-create-time
  %trans-time
  makespan
  product-trans-time
  W1-time-of-processing
  W2-time-of-processing
  already-processed?
  max-pheromones-blue
  max-pheromones-yellow
  maximum-pheromone-level
]

breed [ product ]
breed [ workstation ]

patches-own[
  bifurcacao?
  directions
  pheromones_blue
  pheromones_yellow
  pheromones_W1
  pheromones_W2
  exit?
  ID
  velocity
  availability
```

```
  malfunction?
]

product-own[
  memory
  birth-time
  death-time
  service-list
  orders
  processing
  time-of-processing
  ]

;Cria o Sistema de tapetes.************************
to setup
  ca
  set-patch-size 40
  resize-world -5 5 -5 5
  set tapetes patches with [
    pycor = -1 and pxcor >= -1 and pxcor < 2
    or
    pycor = 1 and pxcor >= -1 and pxcor < 2
    or
    pycor = 0 and pxcor >= -1 and pxcor < 2
    or
    pycor = -4 and pxcor >= -1 and pxcor < 2
    ]
  ask tapetes [
        set pcolor white
        set bifurcacao? TRUE
        set exit? FALSE
    ]
;**************************************
  set lifter1 patches with [
    pxcor = -2 and pycor >= -4 and pycor <= 0
    ]
    ask lifter1 [
        set pcolor grey
        set bifurcacao? TRUE
        set exit? FALSE
        set availability TRUE
    ]
  set lifter2 patches with [
    pxcor = 2 and pycor >= -4 and pycor <= 0
    ]
    ask lifter2 [
        set pcolor grey
        set bifurcacao? TRUE
        set exit? FALSE
        set availability TRUE
    ]
;**************************************
   setup-bifurcacao?
   setup-exit?
   setup-workstation
   set count-box-blue 0
   set count-box-yellow 0
   set OUT-blue 0
   set OUT-yellow 0
   set product-exit-time []
   set product-trans-time []
```

```
end

to setup-plot
  set-current-plot "WIP"
  set-plot-y-range 0 num-box-yellow + num-box-blue
end

to setup-bifurcacao?
   ask patches[ set plabel-color Blue ]
   ask patch -1 1 [
     set directions [-1 -1 180 -1]
     set pheromones_blue  [-1 -1 0 -1]
     set pheromones_yellow  [-1 -1 0 -1]
     set pheromones_W1  [-1 -1 0 -1]
     set pheromones_W2  [-1 -1 0 -1]
     set plabel "C1"
     set velocity 1
     set malfunction? FALSE
     set availability TRUE
     set ID -1
   ]
   ask patch 0 1 [
     set directions [-1 -1 -1 270]
     set pheromones_blue  [-1 -1 -1 0]
     set pheromones_yellow  [-1 -1 -1 0]
     set pheromones_W1  [-1 -1 -1 0]
     set pheromones_W2  [-1 -1 -1 0]
     set plabel "C2"
     set velocity 1
     set malfunction? FALSE
     set availability TRUE
     set ID -1
   ]
   ask patch 1 1 [
     set directions [-1 -1 -1 270]
     set pheromones_blue  [-1 -1 -1 0]
     set pheromones_yellow  [-1 -1 -1 0]
     set pheromones_W1  [-1 -1 -1 0]
     set pheromones_W2  [-1 -1 -1 0]
     set plabel "C3"
     set velocity 1
     set malfunction? FALSE
     set availability TRUE
     set ID -1
   ]
   ask patch -1 0 [
     set directions [-1 90 180 -1]
     set pheromones_blue [-1 0 0 -1]
     set pheromones_yellow [-1 0 0 -1]
     set pheromones_W1 [-1 0 0 -1]
     set pheromones_W2 [-1 0 0 -1]
     set plabel "C4"
     set velocity 1
     set malfunction? FALSE
     set availability TRUE
     set ID -1
   ]
   ask patch 0 0 [
     set directions [-1 90 -1 -1]
     set pheromones_blue  [-1 0 -1 -1]
     set pheromones_yellow  [-1 0 -1 -1]
```

```
  set pheromones_W1  [-1 0 -1 -1]
  set pheromones_W2  [-1 0 -1 -1]
  set plabel "C5"
  set velocity 1
  set malfunction? FALSE
  set availability TRUE
  set ID -1
]
ask patch 1 0 [
  set directions [0 90 -1 -1]
  set pheromones_blue  [0 0 -1 -1]
  set pheromones_yellow  [0 0 -1 -1]
  set pheromones_W1  [0 0 -1 -1]
  set pheromones_W2  [0 0 -1 -1]
  set plabel "C6"
  set velocity 1
  set malfunction? FALSE
  set availability TRUE
  set ID -1
]
ask patch -1 -1 [
  set directions [-1 90 -1 -1]
  set pheromones_blue  [-1 0 -1 -1]
  set pheromones_yellow  [-1 0 -1 -1]
  set pheromones_W1  [-1 0 -1 -1]
  set pheromones_W2  [-1 0 -1 -1]
  set plabel "C7"
  set velocity 1
  set malfunction? FALSE
  set availability TRUE
  set ID -1
]
ask patch 0 -1 [
  set directions [-1 90 -1 -1]
  set pheromones_blue  [-1 0 -1 -1]
  set pheromones_yellow  [-1 0 -1 -1]
  set pheromones_W1  [-1 0 -1 -1]
  set pheromones_W2  [-1 0 -1 -1]
  set plabel "C8"
  set velocity 1
  set malfunction? FALSE
  set availability TRUE
  set ID -1
]
ask patch 1 -1 [
  set directions [0 -1 -1 -1]
  set pheromones_blue  [0 -1 -1 -1]
  set pheromones_yellow  [0 -1 -1 -1]
  set pheromones_W1  [0 -1 -1 -1]
  set pheromones_W2  [0 -1 -1 -1]
  set plabel "C9"
  set velocity 1
  set malfunction? FALSE
  set availability TRUE
  set ID -1
]
ask patch -1 -4 [
  set directions [-1 -1 -1 270]
  set pheromones_blue  [-1 -1 -1 0]
  set pheromones_yellow  [-1 -1 -1 0]
  set pheromones_W1  [-1 -1 -1 0]
```

```
      set pheromones_W2  [-1 -1 -1 0]
      set plabel "C10"
      set velocity 1
      set malfunction? FALSE
      set availability TRUE
      set ID -1
    ]
    ask patch 1 -4 [
      set directions [-1 -1 -1 270]
      set pheromones_blue  [-1 -1 -1 0]
      set pheromones_yellow  [-1 -1 -1 0]
      set pheromones_W1  [-1 -1 -1 0]
      set pheromones_W2  [-1 -1 -1 0]
      set plabel "C11"
      set velocity 2
      set malfunction? FALSE
      set availability TRUE
      set ID -1
    ]
    ask patch -2 0 [
     set directions [-1 90 -1 -1]
     set pheromones_blue  [-1 0 -1 -1]
     set pheromones_yellow  [-1 0 -1 -1]
     set pheromones_W1  [-1 0 -1 -1]
     set pheromones_W2  [-1 0 -1 -1]
     set plabel "L1"
     set velocity 1
     set malfunction? FALSE
     set availability TRUE
     set ID -1
    ]
    ask patch 2 0 [
     set directions [-1 90 180 -1]
     set pheromones_blue  [-1 0 0 -1]
     set pheromones_yellow  [-1 0 0 -1]
     set pheromones_W1  [-1 0 0 -1]
     set pheromones_W2  [-1 0 0 -1]
     set plabel "L2"
     set velocity 1
     set malfunction? FALSE
     set availability TRUE
     set ID -1
    ]
;*************************************
    ask patch -2 -1 [
     set directions [0 -1 -1 -1]
     set pheromones_blue  [0 -1 -1 -1]
     set pheromones_yellow  [0 -1 -1 -1]
     set pheromones_W1  [0 -1 -1 -1]
     set pheromones_W2  [0 -1 -1 -1]
     set velocity 1
     set malfunction? FALSE
     set availability TRUE
     set ID -1
     set plabel "L11"
    ]
    ask patch -2 -2 [
     set directions [0 -1 -1 -1]
     set pheromones_blue  [0 -1 -1 -1]
     set pheromones_yellow  [0 -1 -1 -1]
     set pheromones_W1  [0 -1 -1 -1]
```

```
 set pheromones_W2  [0 -1 -1 -1]
 set velocity 1
 set malfunction? FALSE
 set availability TRUE
 set ID -1
 set plabel "L12"
]
ask patch -2 -3 [
 set directions [0 -1 -1 -1]
 set pheromones_blue  [0 -1 -1 -1]
 set pheromones_yellow  [0 -1 -1 -1]
 set pheromones_W1  [0 -1 -1 -1]
 set pheromones_W2  [0 -1 -1 -1]
 set velocity 1
 set malfunction? FALSE
 set availability TRUE
 set ID -1
 set plabel "L13"
]
ask patch -2 -4 [
 set directions [0 -1 -1 -1]
 set pheromones_blue  [0 -1 -1 -1]
 set pheromones_yellow  [0 -1 -1 -1]
 set pheromones_W1  [0 -1 -1 -1]
 set pheromones_W2  [0 -1 -1 -1]
 set velocity 1
 set malfunction? FALSE
 set availability TRUE
 set ID -1
 set plabel "L14"
]
ask patch 2 -1 [
 set directions [-1 -1 180 -1]
 set pheromones_blue  [-1 -1 0 -1]
 set pheromones_yellow  [-1 -1 0 -1]
 set pheromones_W1  [-1 -1 0 -1]
 set pheromones_W2  [-1 -1 0 -1]
 set velocity 1
 set malfunction? FALSE
 set availability TRUE
 set ID -1
 set plabel "L21"
]
ask patch 2 -2 [
 set directions [-1 -1 180 -1]
 set pheromones_blue  [-1 -1 0 -1]
 set pheromones_yellow  [-1 -1 0 -1]
 set pheromones_W1  [-1 -1 0 -1]
 set pheromones_W2  [-1 -1 0 -1]
 set velocity 1
 set malfunction? FALSE
 set availability TRUE
 set ID -1
 set plabel "L22"
]
ask patch 2 -3 [
 set directions [-1 -1 180 -1]
 set pheromones_blue  [-1 -1 0 -1]
 set pheromones_yellow  [-1 -1 0 -1]
 set pheromones_W1  [-1 -1 0 -1]
 set pheromones_W2  [-1 -1 0 -1]
```

```
    set velocity 1
    set malfunction? FALSE
    set availability TRUE
    set ID -1
    set plabel "L23"
   ]
   ask patch 2 -4 [
    set directions [-1 -1 -1 270]
    set pheromones_blue  [-1 -1 -1 0]
    set pheromones_yellow  [-1 -1 -1 0]
    set pheromones_W1  [-1 -1 -1 0]
    set pheromones_W2  [-1 -1 -1 0]
    set velocity 1
    set malfunction? FALSE
    set availability TRUE
    set ID -1
    set plabel "L24"
   ]

   ask patch 0 -4 [
    set directions [-1 -1 -1 270]
    set pheromones_blue  [-1 -1 -1 0]
    set pheromones_yellow  [-1 -1 -1 0]
    set pheromones_W1  [-1 -1 -1 0]
    set pheromones_W2  [-1 -1 -1 0]
    set velocity 1
   ]
;*************************************************************************
     **********************
   set maximum-pheromone-level 99999
end

to create-box [ box-color ]
 set-default-shape product "box"
 ask patch -3 0 [
   ifelse count product-here = 0 [ set IN-free TRUE ] [ set IN-free
     FALSE ]
 ];end ask patches
 if IN-free and (ticks - last-create-time) >= time-to-create-box [
   create-product 1 [
    setxy -3 0
    set size 0.5
    set heading 90
    set color box-color
    set birth-time ticks
    if color = Blue [ set service-list ["W2" "OUT"]
      set count-box-blue count-box-blue + 1 ]
    if color = Yellow [ set service-list ["W1" "OUT"]
      set count-box-yellow count-box-yellow + 1 ]
    set orders []
    set processing FALSE
    set time-of-processing 0
    set memory []
    ]
   set last-create-time ticks
 ]
end

to setup-exit?
  ask patches[
   if pxcor = -3 and pycor = 0 [
```

```
     set pcolor Green
     set exit? FALSE
;     set ID "IN"
     set bifurcacao? FALSE
     set plabel "IN"
     set directions [-1 90 -1 -1]
     set velocity 1
     set malfunction? FALSE
     set availability TRUE
    set pheromones_blue  [-1 -1 -1 0]
    set pheromones_yellow  [-1 -1 -1 0]
    set pheromones_W1 [-1 -1 -1 0]
    set pheromones_W2  [-1 -1 -1 0]
    set ID -1
   ]
  ]
  ask patches[
   if pxcor = 3 and pycor = 0 [
     set pcolor red
     set exit? TRUE
 ;    set ID "OUT"
     set plabel "OUT"
     set malfunction? FALSE ]
     set availability TRUE
  ]
end

to setup-workstation
   ask patch 0 2 [
;     set directions [0 -1 -1 -1]
     set plabel "W1"
;     set velocity 1
     set pcolor Yellow
   ]
   ask patch 0 -2 [
;     set directions [0 -1 -1 -1]
     set plabel "W2"
;     set velocity 1
     set pcolor Yellow
   ]
set-default-shape workstation "person"
create-workstation 1[
  set size .5
  setxy 0 2
  set color red
]
create-workstation 1[
  set size .5
  setxy 0 -2
  set color red
]
end

to find-exit-random?
set exit-found? FALSE
set aux-directions []
set already-processed? FALSE
 ask product[
   lifters-available?
   ifelse not exit? [
       ask patch-here[
```

```
  set distance-to-jump velocity
  set new_directions remove -1 directions ]
ifelse not bifurcacao? and processing = FALSE [ if not any?
product-on patch-ahead distance-to-jump and not [malfunction?] of
patch-ahead distance-to-jump  and [availability] of patch-ahead
distance-to-jump[ jump distance-to-jump ] ]
[ set list_length length new_directions
  set heading item random list_length new_directions
  if first service-list = "W1" and plabel = "C2" and processing =
FALSE [
     set service-list remove-item 0 service-list
     set processing TRUE
     set time-of-processing time-of-processing + 1
     set color color + 10
     set pcolor 127
      ]
  if first service-list = "W2" and  plabel = "C8" and processing
= FALSE [
       set service-list remove-item 0 service-list
       set processing TRUE
       set time-of-processing time-of-processing + 1
       set color color + 10
       set pcolor 127 ]
 if [plabel] of patch-ahead distance-to-jump = "L1" and plabel !=
"IN" and not processing and not any? product-on patch-ahead
distance-to-jump [ jump distance-to-jump
    set already-processed? TRUE ]
 if [availability] of patch-ahead distance-to-jump [
  if plabel = "IN" and [availability] of patch-ahead distance-to-
jump and not processing and not [malfunction?] of patch-ahead
distance-to-jump and not any? product-on patch-ahead distance-to-
jump [ jump distance-to-jump
    set already-processed? TRUE]
  if plabel = "C10" and [availability] of patch-ahead distance-
to-jump and not processing and not [malfunction?] of patch-ahead
distance-to-jump and not any? product-on patch-ahead distance-to-
jump [jump distance-to-jump
    set already-processed? TRUE ]
  if plabel = "C6" and [availability] of patch-ahead distance-to-
jump and not processing and not [malfunction?] of patch-ahead
distance-to-jump and not any? product-on patch-ahead distance-to-
jump [jump distance-to-jump
    set already-processed? TRUE ]
  if not any? product-on patch-ahead distance-to-jump and not
[malfunction?] of patch-ahead distance-to-jump [
     if [plabel] of patch-ahead distance-to-jump = "L2" and
[availability] of patch-ahead distance-to-jump and not processing
and not any? product-on patch-ahead distance-to-jump [ jump
distance-to-jump
       set already-processed? TRUE ]
     if [plabel] of patch-ahead distance-to-jump = "L1" and
[availability] of patch-ahead distance-to-jump and not processing
and not any? product-on patch-ahead distance-to-jump [ jump
distance-to-jump
       set already-processed? TRUE ] ]
if plabel = "L2" and first service-list != "OUT" and not
processing and not already-processed? and not any? product-on
patch-ahead distance-to-jump [ set heading 180
  jump distance-to-jump
  set already-processed? TRUE ]
```

```
     if plabel = "L2" and first service-list = "OUT" and not
     processing and not already-processed? and not any? product-on
     patch-ahead distance-to-jump [ set heading 90
       jump distance-to-jump
       set already-processed? TRUE ]
     if not any? product-on patch-ahead distance-to-jump and not
     processing and not [malfunction?] of patch-ahead distance-to-jump
     and not already-processed? [ jump distance-to-jump ]
       ]
     ]
   ]
;************ else exit **********
   [ count-exit-box ]
]
end

to find-exit-function?

set P1 [ "C4" "C5" "C6" "C3" "C2" "C1" "C4" ]
set P2 [ "C4" "C7" "C8" "C9" "C6" "C3" "C2" "C1" "C4" ]
set P3 [ "C4" "C5" "C6" "L2" "C11" "C10" "L1" "C4" ]
set P4 [ "C4" "C5" "C6" "L2" "C11" "C10" "L1" "C4" "C5" "C6" "C3" "C2"
     "C1" "C4" ]
set P5 [ "C4" "C5" "C6" "L2" "C11" "C10" "L1" "C4" "C7" "C8" "C9" "C6"
     "C3" "C2" "C1" "C4" ]
set P6 [ "C5" "C6" "L2" "OUT" ]
set P7 [ "L2" "L2D" "L2D" "L2D" "L2D" "C11" "C10" "L1UD" "L1U" "L1U"
     "L1U" "L1" "C4" ]
set P8 [ "C7" "C8" "C9" "C6" "L2" "OUT" ]
set P9 [ "C3" "C2" "C1" "C4"]
set P10 ["C9" "C6" "L2" "OUT" ]
set P11 [ "C1" "C4" ]
set P12 [ "L2" "OUT" ]
set P13 [ "C7" "C8" "C9" "C6" "C3" "C2" "C1" "C4" ]

ask product [
  lifters-available?
  ask patch-here [ set distance-to-jump velocity ]
  if plabel = "IN" and not any? product-on patch-ahead distance-to-
     jump and not processing and [availability] of patch-ahead
     distance-to-jump [
    jump distance-to-jump ]
  if (xcor = -2) and (ycor = -1) [
      set heading 0
      jump distance-to-jump
      set orders remove-item 0 orders
     ]
  if plabel = "L1" and not any? product-on patch-ahead distance-to-
     jump and empty? orders and not processing and [availability] of
     patch-ahead distance-to-jump [
    if first service-list = "W1" [ set orders P1 ]
    if first service-list = "W2" [ set orders P2 ]
    if first service-list = "OUT" [set orders P6 ]
  ]
  if plabel = "C4" and empty? orders and not processing [
     ;ver qual a próxima ordem no service-list e atribuir
    if first service-list = "W1" and not processing [ set orders P1
        set orders remove-item 0 orders ]
    if first service-list = "W2" and not processing [ set orders P2
       set orders remove-item 0 orders ]
    if first service-list = "OUT" and not processing [
```

```
      set heading 90
      ifelse not any? product-on patch-ahead distance-to-jump and not
     processing and [availability] of patch-ahead distance-to-jump [
     set orders P6 ]
      [ set orders P8 ] ]
  ]
  if not empty? orders and not processing [
    next-heading? first orders
    set heading next-heading
    if not any? product-on patch-ahead distance-to-jump and not
     processing and not [malfunction?] of patch-ahead distance-to-jump
     and [availability] of patch-ahead distance-to-jump  [
      jump distance-to-jump
      if first service-list = "W1" and plabel = "C2" [ set processing
     TRUE
         set service-list remove-item 0 service-list
         set color color + 10
         set pcolor 127 ]
      if first service-list = "W2" and plabel = "C8" [ set processing
     TRUE
         set service-list remove-item 0 service-list
         set color color + 10
         set pcolor 127 ]
         set orders remove-item 0 orders
    if any? product-on patch-ahead distance-to-jump and plabel = "C6"
     and last orders = "C4" and not processing and not [malfunction?]
     of patch-ahead distance-to-jump [
      set orders P7 ]
     if plabel = "C6" and not processing and not [malfunction?] of
     patch-ahead distance-to-jump and not [availability] of patch-ahead
     distance-to-jump [
      set orders P9 ]
     if plabel = "C6" and not processing and not [malfunction?] of
     patch-ahead distance-to-jump and [availability] of patch-ahead
     distance-to-jump and first service-list = "OUT" [
       set orders P12 ]
    ]
    ]
  if [malfunction?] of patch-ahead distance-to-jump [
    if first service-list = "OUT" and not processing [ set orders P8 ]
    if first service-list = "W1" and not processing [ set orders P2
      set orders remove-item 0 orders ]
    if first service-list = "W2" and not processing [ set orders P2
      set orders remove-item 0 orders ]
    ]
  if plabel = "C8" and empty? orders and not processing and first
     service-list = "OUT" [ set orders P10 ]
  if plabel = "C2" and empty? orders and not processing and first
     service-list = "OUT" [ set orders P11 ]

  ]
count-exit-box
end

to find-exit-pheromones?
set exit-found? FALSE
set aux-directions []
set already-processed? FALSE
 ask product[
   let aux-scent []
   lifters-available?
```

```
ifelse not exit? [
    ask patch-here[
    set distance-to-jump velocity
    set max-pheromones-blue max pheromones_blue
    set max-pheromones-yellow max pheromones_yellow
     ]
  ifelse not bifurcacao? and not processing [ if not any? product-
  on patch-ahead distance-to-jump and not [malfunction?] of patch-
  ahead distance-to-jump and [availability] of patch-ahead
  distance-to-jump[ jump distance-to-jump ] ]
  [ if not processing [
    ifelse (first service-list = "W1") or (first service-list =
  "W2") [
    ifelse (color = Blue) or (color = Blue + 10) and (max
  pheromones_W2) > pheromones-decision-level [ set heading item
  (position max pheromones_W2 pheromones_W2) directions
      let labelp [plabel] of patch-here
      let scent-position item (position (max pheromones_W2)
  pheromones_W2) directions
      set aux-scent lput labelp aux-scent
      set aux-scent lput scent-position aux-scent
      set memory lput aux-scent  memory ]
   [ ifelse (color = Yellow) or (color = Yellow + 10) and (max
  pheromones_W1) > pheromones-decision-level [ set heading item
  (position (max pheromones_W1) pheromones_W1) directions
      let labelp [plabel] of patch-here
      let scent-position item (position max pheromones_W1
  pheromones_W1) directions
      set aux-scent lput labelp aux-scent
      set aux-scent lput scent-position aux-scent
      set memory lput aux-scent  memory ]
   [ set new_directions remove -1 directions
     set list_length length new_directions
     set heading item random list_length new_directions
     let labelp [plabel] of patch-here
     let scent-position heading
     set aux-scent lput labelp aux-scent
     set aux-scent lput scent-position aux-scent
     set memory lput aux-scent  memory ]  ] ]
    [ifelse (color = Blue) or (color = Blue + 10) and (max
  pheromones_blue) > pheromones-decision-level [ set heading item
  (position max pheromones_blue pheromones_blue) directions
      let labelp [plabel] of patch-here
      let scent-position item (position (max pheromones_blue)
  pheromones_blue) directions
      set aux-scent lput labelp aux-scent
      set aux-scent lput scent-position aux-scent
      set memory lput aux-scent  memory ]
   [ ifelse (color = Yellow) or (color = Yellow + 10) and (max
  pheromones_yellow) > pheromones-decision-level [ set heading item
  (position (max pheromones_yellow) pheromones_yellow) directions
      let labelp [plabel] of patch-here
      let scent-position item (position max-pheromones-yellow
  pheromones_yellow) directions
      set aux-scent lput labelp aux-scent
      set aux-scent lput scent-position aux-scent
      set memory lput aux-scent  memory ]
   [ set new_directions remove -1 directions
     set list_length length new_directions
     set heading item random list_length new_directions
     let labelp [plabel] of patch-here
```

```
    let scent-position heading
    set aux-scent lput labelp aux-scent
    set aux-scent lput scent-position aux-scent
    set memory lput aux-scent  memory ]  ]  ]
  if  first  service-list  =  "W1"  and  plabel  =  "C2"  and  not
processing [
     set service-list remove-item 0 service-list
     set processing TRUE
     set time-of-processing time-of-processing + 1
     set color color + 10
     set pcolor 127
     update-pheromones-to-workstation-W1 ]
  if  first  service-list  =  "W2"  and   plabel  =  "C8"  and  not
processing [
       set service-list remove-item 0 service-list
       set processing TRUE
       set time-of-processing time-of-processing + 1
       set color color + 10
       set pcolor 127
       update-pheromones-to-workstation-W2 ]
 if [plabel] of patch-ahead distance-to-jump = "L1" and plabel !=
"IN"  and  not  processing  and  not  any?  product-on  patch-ahead
distance-to-jump [ jump distance-to-jump
    set already-processed? TRUE ]
 if [availability] of patch-ahead distance-to-jump [
  if plabel = "IN" and [availability] of patch-ahead distance-to-
jump and not processing and not [malfunction?] of patch-ahead
distance-to-jump and not any? product-on patch-ahead distance-to-
jump [ jump distance-to-jump
    set already-processed? TRUE]
  if plabel = "C10" and [availability] of patch-ahead distance-
to-jump and not processing and not [malfunction?] of patch-ahead
distance-to-jump and not any? product-on patch-ahead distance-to-
jump [jump distance-to-jump
    set already-processed? TRUE ]
  if plabel = "C6" and [availability] of patch-ahead distance-to-
jump and not processing and not [malfunction?] of patch-ahead
distance-to-jump and not any? product-on patch-ahead distance-to-
jump [jump distance-to-jump
    set already-processed? TRUE ]
  if  not  any?  product-on  patch-ahead  distance-to-jump  and  not
[malfunction?] of patch-ahead distance-to-jump [
     if  [plabel]  of  patch-ahead  distance-to-jump  =  "L2"  and
[availability] of patch-ahead distance-to-jump and not processing
and  not  any?  product-on  patch-ahead  distance-to-jump  [  jump
distance-to-jump
       set already-processed? TRUE ]
     if  [plabel]  of  patch-ahead  distance-to-jump  =  "L1"  and
[availability] of patch-ahead distance-to-jump and not processing
and  not  any?  product-on  patch-ahead  distance-to-jump  [  jump
distance-to-jump
       set already-processed? TRUE ] ]
if  plabel  =  "L2"  and  first  service-list  !=  "OUT"  and  not
processing and not already-processed? and not any? product-on
patch-ahead distance-to-jump [ set heading 180
  jump distance-to-jump
  set already-processed? TRUE ]
if  plabel  =  "L2"  and  first  service-list  =  "OUT"  and  not
processing and not already-processed? and not any? product-on
patch-ahead distance-to-jump [ set heading 90
  jump distance-to-jump
```

```
      set already-processed? TRUE ]
     if plabel = "C6" and first service-list != "OUT" and not
    processing and not already-processed? and not any? product-on
    patch-ahead distance-to-jump [ set heading 180
      jump distance-to-jump
      set already-processed? TRUE ]
    if not any? product-on patch-ahead distance-to-jump and not
    processing and not [malfunction?] of patch-ahead distance-to-jump
    and not already-processed? [ jump distance-to-jump ]
      ]
     if not already-processed? and [malfunction?] of patch-ahead
    distance-to-jump [
      let temp position heading  directions
      set pheromones_W1 replace-item temp pheromones_W1 0
      set pheromones_W2 replace-item temp pheromones_W2 0
      set pheromones_blue replace-item temp pheromones_blue 0
      set pheromones_yellow replace-item temp pheromones_yellow 0
       ]
    ]
    ]
  ]
  [ count-exit-box ]
]
end

to next-heading? [  final ]
 set heading 90
 if [plabel] of patch-ahead distance-to-jump = final [ set next-
    heading 90 ]
 set heading 0
 if [plabel] of patch-ahead distance-to-jump = final [ set next-
    heading 0 ]
 set heading 270
 if [plabel] of patch-ahead distance-to-jump = final [ set next-
    heading 270 ]
 set heading 180
 if [plabel] of patch-ahead distance-to-jump = final [ set next-
    heading 180 ]
 if final = "L1U" [ set next-heading 0 ]
 if final = "L2D" [ set next-heading 180 ]
 if final = "L1UD" [set next-heading 270 ]
end

to update-pheromones-to-workstation-W1
  set memory remove-duplicates memory
  let life-time (ticks - birth-time)
  foreach memory [ let temp first memory
     let patch-label first temp
     let patch-heading last temp
     ask patches with [ plabel = patch-label and ID = -1] [ let temp2
    position patch-heading directions
     ifelse ((item temp2 pheromones_W1) + (pheromone-deposition * (8
    + W1-time-of-processing) / life-time )) > maximum-pheromone-level
    [
           set pheromones_W1 replace-item temp2 pheromones_W1
    maximum-pheromone-level ]
           [ set pheromones_W1 replace-item temp2 pheromones_W1
    ((item temp2 pheromones_W1) + (pheromone-deposition * (8 + W1-
    time-of-processing) / life-time )) ]  ]
     set memory remove-item 0 memory]
 set memory []
```

```
end

to update-pheromones-to-workstation-W2
   set memory remove-duplicates memory
   let life-time (ticks - birth-time)
   foreach memory [ let temp first memory
      let patch-label first temp
      let patch-heading last temp
      ask patches with [ plabel = patch-label and ID = -1] [ let temp2
     position patch-heading directions
      ifelse ((item temp2 pheromones_W2) + (pheromone-deposition * (4
     + W2-time-of-processing) / life-time )) > maximum-pheromone-level
     [
        set pheromones_W2 replace-item temp2 pheromones_W2 maximum-
     pheromone-level ]
        [ set pheromones_W2 replace-item temp2 pheromones_W2 ((item
     temp2 pheromones_W2) + (pheromone-deposition * (4 + W2-time-of-
     processing) / life-time )) ]  ]
      set memory remove-item 0 memory]
  set memory []
end


to count-exit-box
ask product [
  if  plabel = "OUT" [ if color = Blue + 10 [  set OUT-blue OUT-blue +
     1
      set death-time ticks
      let life-time (death-time - birth-time)
      set product-exit-time lput (death-time - birth-time) product-
     exit-time
      set product-trans-time lput (death-time - birth-time - blue-
     time-of-processing) product-trans-time
      if method = "Pheromones" [
        set memory remove-duplicates memory
        foreach memory [ let temp first memory
          let patch-label first temp
          let patch-heading last temp
          ask patches with [ plabel = patch-label and ID = -1] [ let
     temp2 position patch-heading directions
          ifelse ((item temp2 pheromones_blue) + (pheromone-deposition
     *  (8  +  blue-time-of-processing)  /  life-time  )) > maximum-
     pheromone-level [
            set pheromones_blue replace-item temp2 pheromones_blue
     maximum-pheromone-level ]
            [ set pheromones_blue replace-item temp2 pheromones_blue
     ((item temp2 pheromones_blue) + (pheromone-deposition * (8 + blue-
     time-of-processing) / life-time )) ]  ]
          set memory remove-item 0 memory]
        ]
        die  ]
    if color = Yellow + 10 [  set OUT-yellow OUT-yellow + 1
        set death-time ticks
        let life-time (death-time - birth-time)
        set product-exit-time lput (death-time - birth-time) product-
     exit-time
        set product-trans-time lput (death-time - birth-time - yellow-
     time-of-processing) product-trans-time
      if method = "Pheromones" [
        set memory remove-duplicates memory
        foreach memory [ let temp first memory
```

```
          let patch-label first temp
          let patch-heading last temp
          ask patches with [ plabel = patch-label and ID = -1 ] [ let
     temp2 position patch-heading directions
          ifelse   ((item   temp2   pheromones_yellow)   +   (pheromone-
     deposition * (12 + yellow-time-of-processing) / life-time )) >
     maximum-pheromone-level [
            set pheromones_yellow replace-item temp2 pheromones_yellow
     maximum-pheromone-level ]
            [     set     pheromones_yellow     replace-item     temp2
     pheromones_yellow ((item temp2 pheromones_yellow) + (pheromone-
     deposition * (12 + yellow-time-of-processing) / life-time )) ]  ]
          set memory remove-item 0 memory]
        ]
        die ]
        ]
]
end

to evaporation
  ask tapetes [
    foreach pheromones_blue [ if ? != -1 [ ifelse (? - evaporation-
     level)  >  0  [  set  pheromones_blue  replace-item  (position  ?
     pheromones_blue) pheromones_blue (? - evaporation-level) ]
         [    set    pheromones_blue    replace-item    (position    ?
     pheromones_blue) pheromones_blue 0 ] ] ]
    foreach pheromones_yellow [ if ? != -1 [ ifelse (? - evaporation-
     level)  >  0  [  set  pheromones_yellow  replace-item  (position  ?
     pheromones_yellow) pheromones_yellow (? - evaporation-level) ]
         [    set    pheromones_yellow    replace-item    (position    ?
     pheromones_yellow) pheromones_yellow 0 ] ] ]
     foreach pheromones_W1 [ if ? != -1 [ ifelse (? - evaporation-
     level)  >  0  [  set  pheromones_W1  replace-item  (position  ?
     pheromones_W1) pheromones_W1 (? - evaporation-level) ]
         [ set pheromones_W1 replace-item (position ? pheromones_W1)
     pheromones_W1 0 ] ] ]
     foreach pheromones_W2 [ if ? != -1 [ ifelse (? - evaporation-
     level)  >  0  [  set  pheromones_W2  replace-item  (position  ?
     pheromones_W2) pheromones_W2 (? - evaporation-level) ]
         [ set pheromones_W2 replace-item (position ? pheromones_W2)
     pheromones_W2 0 ] ] ]
     ]
    ask lifter1 [
    foreach pheromones_blue [ if ? != -1 [ ifelse (? - evaporation-
     level)  >  0  [  set  pheromones_blue  replace-item  (position  ?
     pheromones_blue) pheromones_blue (? - evaporation-level) ]
         [    set    pheromones_blue    replace-item    (position    ?
     pheromones_blue) pheromones_blue 0 ] ] ]
    foreach pheromones_yellow [ if ? != -1 [ ifelse (? - evaporation-
     level)  >  0  [  set  pheromones_yellow  replace-item  (position  ?
     pheromones_yellow) pheromones_yellow (? - evaporation-level) ]
         [    set    pheromones_yellow    replace-item    (position    ?
     pheromones_yellow) pheromones_yellow 0 ] ] ]
     foreach pheromones_W1 [ if ? != -1 [ ifelse (? - evaporation-
     level)  >  0  [  set  pheromones_W1  replace-item  (position  ?
     pheromones_W1) pheromones_W1 (? - evaporation-level) ]
         [ set pheromones_W1 replace-item (position ? pheromones_W1)
     pheromones_W1 0 ] ] ]
     foreach pheromones_W2 [ if ? != -1 [ ifelse (? - evaporation-
     level)  >  0  [  set  pheromones_W2  replace-item  (position  ?
     pheromones_W2) pheromones_W2 (? - evaporation-level) ]
```

```
        [ set pheromones_W2 replace-item (position ? pheromones_W2)
     pheromones_W2 0 ] ] ]
     ]
    ask lifter2 [
    foreach pheromones_blue [ if ? != -1 [ ifelse (? - evaporation-
     level) > 0 [ set pheromones_blue replace-item (position ?
     pheromones_blue) pheromones_blue (? - evaporation-level) ]
        [ set pheromones_blue replace-item (position ?
     pheromones_blue) pheromones_blue 0 ] ] ]
    foreach pheromones_yellow [ if ? != -1 [ ifelse (? - evaporation-
     level) > 0 [ set pheromones_yellow replace-item (position ?
     pheromones_yellow) pheromones_yellow (? - evaporation-level) ]
        [ set pheromones_yellow replace-item (position ?
     pheromones_yellow) pheromones_yellow 0 ] ] ]
     foreach pheromones_W1 [ if ? != -1 [ ifelse (? - evaporation-
     level) > 0 [ set pheromones_W1 replace-item (position ?
     pheromones_W1) pheromones_W1 (? - evaporation-level) ]
        [ set pheromones_W1 replace-item (position ? pheromones_W1)
     pheromones_W1 0 ] ] ]
     foreach pheromones_W2 [ if ? != -1 [ ifelse (? - evaporation-
     level) > 0 [ set pheromones_W2 replace-item (position ?
     pheromones_W2) pheromones_W2 (? - evaporation-level) ]
        [ set pheromones_W2 replace-item (position ? pheromones_W2)
     pheromones_W2 0 ] ] ]
     ]
end

to lifters-available?
 ifelse (count product-on lifter1) >= 1 [ ask patch -2 0 [ set
     availability FALSE ] ask patch -2 -4 [ set availability FALSE ] ]
 [ ask patch -2 0 [ set availability TRUE ] ask patch -2 -4 [ set
     availability TRUE ] ]
 ifelse (count product-on lifter2) >= 1 [ ask patch 2 0 [ set
     availability FALSE ] ]
  [ ask patch 2 0 [ set availability TRUE ] ]
end

to calculate-average-time
  set average-product-exit-time mean product-exit-time
  set %trans-time ( mean product-trans-time ) / average-product-
     exit-time * 100
  set makespan ticks
end

to check-availability
  ask patches [
    ifelse count product-here >= 1 [ set availability FALSE ]
    [set availability TRUE]
    ]
end

to machine-processing
  ask product [
    if color = Blue + 10 [
      if processing = TRUE and time-of-processing <= blue-time-of-
     processing [ set time-of-processing time-of-processing + 1
                  set W2-time-of-processing W2-time-of-processing + 1
     ]
      if processing = TRUE and time-of-processing > blue-time-of-
     processing [ set processing FALSE
                  set pcolor White ] ]
```

```
    if color = Yellow + 10 [
      if processing = TRUE and time-of-processing <=  yellow-time-of-
     processing [ set time-of-processing time-of-processing + 1
                  set W1-time-of-processing    W1-time-of-processing + 1
     ]
      if processing = TRUE and time-of-processing >  yellow-time-of-
     processing [ set processing FALSE
                  set pcolor White ] ]
    ]
end

to check-malfunction?
  ask patches[ ifelse plabel = Malfunction [ set malfunction? TRUE ]
    [set malfunction? FALSE ] ]
end

to update-plot
 set-current-plot "WIP"
 set-current-plot-pen "WIP"
 plot count product-on patches
end

to executar
  if ticks >= time-of-malfunction and time-of-malfunction != 0 [ set
     Malfunction "C5"  ]
    ifelse random 2 = 0 [ if count-box-blue < num-box-blue and  OUT-
     blue != num-box-blue [ create-box Blue ] ]
    [ if count-box-yellow < num-box-yellow and  OUT-yellow != num-box-
     yellow [ create-box Yellow ] ]
    check-malfunction?
    machine-processing
    check-availability
    if method = "Random" [find-exit-random?]
    if method = "T-invariant" [find-exit-function?]
    if method = "Pheromones" [find-exit-pheromones?
      evaporation ]
    tick
    update-plot
    if OUT-blue = num-box-blue and OUT-yellow = num-box-yellow [
      calculate-average-time
      export-all-plots "d:/plots.csv"
      stop
      ]
end
```