Selection Constructive based Hyper-heuristic for Dynamic Scheduling

Sílvia Raquel Pinto Gomes

Dissertation to obtain the Master of Science degree in Computer Science, Specialization in Graphic Systems and Multimedia

Supervisor: Ana Madureira

Co-Supervisor: João Paulo Pereira

Jury:

Chairman:

Dr. Maria de Fátima Coutinho Rodrigues

Vowels:

Dr. Maria Leonilde Rocha Varela

Dr. Ana Maria Dias Madureira Pereira

Dr. João Paulo Jorge Pereira

Porto, October 2014

Resumo

A função de escalonamento desempenha um papel importante nos sistemas de produção. Os sistemas de escalonamento têm como objetivo gerar um plano de escalonamento que permite gerir de uma forma eficiente um conjunto de tarefas que necessitam de ser executadas no mesmo período de tempo pelos mesmos recursos. Contudo, adaptação dinâmica e otimização é uma necessidade crítica em sistemas de escalonamento, uma vez que as organizações de produção têm uma natureza dinâmica. Nestas organizações ocorrem distúrbios nas condições requisitos de trabalho regularmente e de forma inesperada. Alguns exemplos destes distúrbios são: surgimento de uma nova tarefa, cancelamento de uma tarefa, alteração na data de entrega, entre outros. Estes eventos dinâmicos devem ser tidos em conta, uma vez que podem influenciar o plano criado, tornando-o ineficiente. Portanto, ambientes de produção necessitam de resposta imediata para estes eventos, usando um método de reescalonamento em tempo real, para minimizar o efeito destes eventos dinâmicos no sistema de produção. Deste modo, os sistemas de escalonamento devem de uma forma automática e inteligente, ser capazes de adaptar o plano de escalonamento que a organização está a seguir aos eventos inesperados em tempo real.

Esta dissertação aborda o problema de incorporar novas tarefas num plano de escalonamento já existente. Deste modo, é proposta uma abordagem de otimização – Hiper-heurística baseada em Seleção Construtiva para Escalonamento Dinâmico- para lidar com eventos dinâmicos que podem ocorrer num ambiente de produção, a fim de manter o plano de escalonamento, o mais robusto possível. Esta abordagem é inspirada em computação evolutiva e hiper-heurísticas. Do estudo computacional realizado foi possível concluir que o uso da hiper-heurística de seleção construtiva pode ser vantajoso na resolução de problemas de otimização de adaptação dinâmica.

Palavras-chave: Ambientes Inteligentes, Escalonamento dinâmico, Hiper-heurísticas, Aprendizagem, Otimização.

Abstract

Scheduling plays an important role in manufacturing systems. It produces a scheduling plan, in order to share resources to produce several different products in the same time period. However, dynamic adaptation and optimization is a critical need in real-world manufacturing scheduling systems, since contemporary manufacturing organizations have a dynamic nature, where disturbances on working conditions and requirements occur on a continuous basis. Disturbances often arise unexpectedly, and can be for example: urgent job arrival, job cancelation, due date change, delay in the arrival, among others. These dynamic events must be taken into account, since they may have a major impact on the scheduling plan, they can disorder the plan making it ineffective. Therefore, manufacturing environments require immediate response to these dynamic events, using a real-time rescheduling method, in order to minimize the effect of such unexpected events in the performance of the production' system. As result, scheduling systems should have the ability of automatically and intelligently maintain real-time adaptation and optimization to efficiently update the scheduling plan to the unexpected events. This way, the organization keeps clients satisfied and achieves its objectives (costs minimized and profits maximized).

This dissertation addresses the problem of incorporating new tasks in a scheduling plan already generated by the scheduling system. Therefore, it proposes an optimization approach - Selection Constructive based Hyper-heuristic for Dynamic Scheduling - to deal with dynamic events that can occur over time in a manufacturing environment, with the main goal of maintaining the current scheduling plan feasible and most robust as possible. The development of this dynamic adaptation approach is inspired on evolutionary computation and hyper-heuristics. The viability of the proposed approach is tested by performing a set of experiments and analysing the results achieved. From the obtained results it is possible to conclude that the use of a selection constructive hyper-heuristic could be advantageous on solving dynamic adaptation optimization problems.

Keywords: Ambient Intelligence, Dynamic Scheduling, Hyper-heuristics, Learning, Optimization

Acknowledgements

Firstly, I would like to thank all the people who somehow supported me during this work and contributed to its success.

I would like to show my gratitude to both my supervisors, Ana Madureira and João Paulo Pereira, for all their assistance along the course of this work. I must say that they were rather tireless in their support.

Further, I would like to thank André Catita, for the all the help provided, and to my supervisor Professor Ana Madureira, for the opportunity and for being always available to clarify doubts and help me developing this project.

Table of Contents

List of Figures

List of Tables

Acronyms and Nomenclature

List of Acronyms

ADSyS	Adaptive Decision Support System for Interactive Scheduling with MetaCognition and User Modeling Experience
ALADIN	Ambient Lighting Assistance for an Ageing Population
AmI	Ambient Intelligence
CALO	Cognitive Agent that Learns and Organizes
EA	Evolutionary Algorithm
EC	Evolutionary Computation
EDD	Earliest Due Date
EP	Evolutionary Programming
ERD	Earliest Release Date
ES	Evolution Strategies
FIFO	First In First Out
GA	Genetic Algorithm
GECAD	Knowledge Engineering and Decision Support research Centre
GP	Genetic Programming
GPF	Greatest Priority First
GPS	Global Positioning System
HCI	Human-Computer Interaction
HH	Hyper-Heuristic
HMM	Hidden Markov Model
IBM	International Business Machines Corporation
ICTs	Information and Communication Technologies
I-EMS	Intelligent Electronic Mail Sorter

IM	Integration Mechanism
iRoom	Interactive Room
ISO	International Organization for Standardization
ISTAG	Information Society Technology Advisory Group
JSP	Job-Shop scheduling Problem
MAS	Multi-Agent System
MIT	Massachusetts Institute of Technology
ML	Machine Learning
MLP	Multilayer Perceptron
OSGi	Open Services Gateway Initiative
PC	Personal Computer
PDA	Personal Digital Assistants
RCSM	Reconfigurable Context Sensitive Middleware
RF	Radio-Frequency
RFID	Radio-Frequency Identification
RWP	Real-World Scheduling Problems
SVM	Support Vector Machine
UAN	User Action Notation
UML	Unified Modeling Language
URL	Uniform Resource Locator
UseML	Useware Markup Language
UsiXML	User Interface Extensible Markup language
WIMP	Windows, Icons, Menus, Pointing device

1 Introduction

This document and the project to which it refers was carried out for the Dissertation of the 2nd year of the Master's Degree in Informatics Engineering of the Department of Informatics, ISEP, This project is also associated with the development of R&D project ADSyS (Adaptive Decision Support System for Interactive Scheduling with MetaCognition and User Modeling Experience) (PTDC/EME-GIN/109956/2009), which is a dynamic scheduling system to solve scheduling problems subject to dynamic events. This project was developed at the research group GECAD (Knowledge Engineering and Decision Support research Centre).
Scheduling plays an important role in manufacturing systems. It creates a scheduling plan, in order to share resources to produce several different products in the same time period, in a way that the organization optimizes its objectives and achieves its goals. Generally, these objectives can be: minimize the time to complete all jobs, minimize the number of jobs that are completed after its due dates, or maximize the occupation of resources, among many others. Overall, a detailed schedule plan helps the organizations maintain efficiency and control of their operations. However daily activity of current manufacturing organizations have a dynamic nature, they constantly have unexpected events and disturbances on working conditions and requirements. These dynamic events and disturbances can be for example: a busy or broke machine, energy sources shortage, urgent job arrival, job cancelation, due date change, delay in the arrival, shortage of materials, and change in job priority, among others.

Dynamic events must be taken into account, since they may have a major impact and can disorder the scheduling plan that the organization is already following, making it ineffective. As result, it can affect the production' performance and compromise the organization' profits. Therefore, manufacturing systems require immediate response to these dynamic events, using a real-time scheduling method. Thus, scheduling systems should have an efficient reactivity to the changing environment, in order to minimize the effect of such unexpected events in the performance of the current scheduling plan.

This master thesis has as major goal of addressing the problem of incorporating dynamic events in a current scheduling plan, in order to maintain the plan feasible, and the organization keeps clients satisfied and achieves its objectives (costs minimized and profit

maximized). Therefore, this work validates the contribution of different artificial intelligence research areas to propose a new approach to address this problem, and compare it with the supervised machine learning approach already implemented in the ADSyS system.

1.1 Main Goals

Considering that ambient intelligence is a new challenge for artificial intelligence, the work developed and described in this master thesis intends to implement a dynamic integration module by combining and integrating different artificial intelligence research areas in order to solve Real-World Scheduling problems (RWP):

- Intelligent Interaction between the user and the DSS which should support the users to access, manage, share and exchange information;
- Ambient Intelligence emphasizes user-friendliness systems, more efficient support services, user-empowerment, and support for human interactions;
- Active Learning provides the ability to learn from users and to infer their intentions. To achieve this goal, some machine learning techniques appear as candidates since it allows the inclusion of knowledge from the domain and is adequate for observational learning;
- Adaptive Optimization Methods incorporating knowledge by user observation and reacting to perturbations from external environment;
- Hybridization of Intelligent Systems is a promising research field of computational intelligence focusing on combinations of multiple approaches to develop the next generation of IS due to their capabilities in handling RWP complexities involving imprecision, uncertainty and vagueness.

What this thesis addresses is incorporated in a R&D project with emphasis on the design of scheduling support systems for manufacturing environments where dynamic adaptation and optimization become increasingly important in incorporating expert's knowledge from ambient intelligence perspective. This master thesis proposes a dynamic integration mechanism module based on selection constructive hyper-heuristics to allow the ADSyS scheduling system to autonomously perform the integration of dynamic events. A hyper-heuristic is considered, since it proved to be useful in ambient intelligence systems (context-aware and human centred domains).

1.2 Main Contributions

This document provides a new approach to dynamically integrate unexpected events in a scheduling plan, using a selection constructive based hyper-heuristic. Experiments were also performed in order to validate the ability of this approach in performing the integration in an

efficient manner, when compared against the supervised machine learning approach already implemented in the ADSyS system.

In short, this work contributes with:

- A implementation of a selection constructive hyper-heuristic, based on an evolutionary algorithm, in order to integrate a dynamic event in the scheduling plan at hand;
- Experiments concerning this proposed approach and to the machine learning approach;
- Validation of the ability of this approach in automatically and efficiently adapt , on a continuous basis, the current schedule to the dynamic events that occur over time;
- Comparison between the results obtained via the two different approaches in order to assess which of them is better.

1.3 Document Structure

This thesis is comprised of 9 chapters which are organized as follows:

- This current chapter presents a description of the work developed, including the motivation, main goals and contributions;
- Chapter 2 presents the state-of-the-art on the major field of this work, artificial intelligence, which serves as background of this work;
- Chapter 3 presents the state-of-the-art on the human-computer-interaction field, in order to focus the importance of the user interfaces in any computational system;
- Chapter 4 presents and analyses an important subarea of artificial intelligence, machine learning, describing the different types of learning and their techniques;
- Chapter 5 assesses the dynamic scheduling problem. First is described the objectives of scheduling in current manufacturing organizations, as well as, the dynamism existent in them, and different methods to deal with it. In addition, this chapter provides a nomenclature that will be used in the remaining of the document while presenting relevant concepts for understanding the work developed;
- Chapter 6 addresses another artificial intelligence subarea, evolutionary computation and hyper-heuristics. This chapter explores this area and describes the major utilized concepts, as well as, the different methods that this subarea encompasses;
- Chapter 7 presents the selection constructive hyper-heuristic proposed in this work, as well as, the implementation and parameterization of the algorithm. In addition, this chapter describes the ADSyS system, the different modules that comprise it and the multi-agent architecture implemented;

- Chapter 8 presents the experiments performed and the results obtained in order to validate the proposed dynamic adaptation approach. A detailed explanation of the experimental setting is provided. The results are presented and discussed in comparison with the machine learning approach;
- Finally chapter 9 presents the conclusions which can be taken from this work and the open issues. It also presents a description of possible future work to further improve the techniques described.

2 Ambient Intelligence

Ambient intelligence (AmI) concept was first introduced by the European Commission's Information Society Technologies Advisory Group (ISTAG) (Weyrich, 2000). The interest in Aml environments has grown considerably due to new challenges posed by society, giving place to new interesting associated researches. Ambient intelligence is an emerging discipline with the vision of future in which environments support the users inhabiting them. The envisioned environment should be interconnected, unobtrusive, dynamic, adaptable, embedded and intelligent (Sadri, 2011). In an Aml scenario the traditional computer input and output media does not exist, instead processors and sensors devices are embedded in all kind of everyday objects (like clothes, vehicles, furniture, roads, even smart materials like particles of decorative substances like paint). These electronic environments where users are surrounded by intelligent and intuitive interfaces that make it possible to bring intelligence to our everyday (Cook, Augusto, & Jakkula, 2009). This chapter will provide a survey of the field of Ambient Intelligence, including its applications, some of the technologies it uses, as well as its social and ethical implications.

2.1 Definitions: Concepts and Paradigms

The evolution of technology was a significant factor in the emergence of the field of Ambient Intelligence, which was what led the European Commission to chart a path for Aml research in 2001, when one of its Programme Advisory Groups, the European Community's Information Society Technology, launched the Aml challenge (Cook et al., 2009; Sadri, 2011). Initially the computers were difficult to understand and use and also a rare resource making it very expensive. A single computer was used by many individuals, today as industry progressed and costs have decreased, the computational resources that we have at our disposal have grown significantly. Nowadays having access more than one computer does not means that people only own a PC (Personal Computer) and a laptop. The miniaturization of microprocessors led

computers to be embedded in objects of our daily life, such as: washing machines, refrigerators, microwave ovens, mobile phones, PDAs (Personal Digital Assistants), cars and GPS (Global Positioning System) navigation, among others. Computers that have the objective of accomplishing specific tasks with a faster computation performance and with reduced use of power are gradually spreading through the today society. This widespread availability of computer resources led the realization of Ambient Intelligence. The major objective of Aml is to support people in their daily lives through digital environments that are sensitive and responsive to the presence of people (Commission, 2001; Cook et al., 2009).

It is possible to create an environment that takes into account the personal requirements and preferences of the users and is capable of anticipating their needs and behaviours, and even interact with them in a user-friendly way (Bick & Kummer, 2010; Sadri, 2011).

Briefly, in a fully developed Aml landscape, a system can be built so that it senses features of the people and their environment (through sensors and devices interconnected through a network), then it reasons about the obtained data using a variety of Ambient Intelligence techniques, and finally reacts according to what will be beneficial for the users' in that environment in a seamless, unobtrusive and often invisible way, this is the basic idea behind Aml. In this context unobtrusive means that the interaction should be enjoyable and relaxing for the user, and should not involve a steep learning curve (Commission, 2001; Cook et al., 2009; Intelligence, n.d.).

ISTAG (Intelligence, n.d.) rejected a concrete definition for Aml in order to avoid restricting the concept (Bick & Kummer, 2010). As result, researches have been characterizing Ambient Intelligence in different ways:

- *"A potential future in which we will be surrounded by intelligent objects and in which the environment will recognize the presence of persons and will respond to it in an undetectable manner."* (Commission, 2001)
- *"A digital environment that supports people in their daily lives in a nonintrusive way (Raffler)."* (Phillips, 2007)
- *"Ambient Intelligence implies intelligence that is all around us."* (Maeda & Minami, 2006)
- *"In an AmI environment people are surrounded with networks of embedded intelligent devices that can sense their state, anticipate, and perhaps adapt to their needs."* (Vasilakos & Pedrycz, 2006)

These definitions highlight the six different features that are generally expected in Ambient Intelligence technologies: sensitive, responsive, adaptive, transparent, ubiquitous, and intelligent. Through these definitions and features it can be seen the common characteristics between Aml and related concepts, such as: pervasive computing, ubiquitous computing and artificial intelligence (Bick & Kummer, 2010; Cook et al., 2009). In the next table it is presented a brief distinction of these concepts, with the use of short examples.

Table 1 – Distinction of the concept ambient intelligence, adapted from (Bick & Kummer, 2010).

Concept	Origin	Focal point
Ambient intelligence	(Weyrich, 2000)	Devices linked by a network that collects and analyses data, and acts according to it.
Ubiquitous computing	(Weiser, 1991b)	Several invisible computers are linked and the related virtual and real words are merged.
Pervasive computing	(Ark & Selker, 1999); (Hansmann, et al., 2001)	Remote computers provide permanent access to information; and the ability to respond accordingly is made possible.

Both Aml and ubiquitous computing have a high degree integration within the environment. But, ubiquitous computing is more focused on specific technical implementation, whereas in Aml the main focus is the users and their needs (Bick & Kummer, 2010).

The fact that in Aml systems are expected features such as: sensitive, responsive and adaptive, highlights the dependence that Aml research has on context-aware computing (Cook et al., 2009). Likewise, the transparent feature is related with the concept of disappearing computer that was envisioned by Weiser, who stated:

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." (Weiser, 1991a).

The disappearing computer concept is directly connected with the concept of Ubiquitous Computing, or Pervasive Computing like it was called by IBM (International Business Machines Corporation), also some technical publications equate it with Ambient Intelligence. A central unique characteristic of Aml when compared with these concepts is the use of artificial analytic intelligence. This means that, the system is sensitive to context, is adaptive, can learn from the behaviour of the users, can act in an independent way and can even recognize emotion (Bick & Kummer, 2010; Sadri, 2011). Thus, Aml should include artificial intelligence research into its purview, containing contributions from machine learning, agent-based software and robotics. Also, Aml research can integrate work on characteristics related with human intelligence, such as hearing, vision, language and knowledge, which is what differs Aml from ubiquitous computing (Cook et al., 2009).

A key aspect for the acceptance of Aml is that it needs to be driven by humanistic concerns, and not only technological determined ones (Commission, 2001; Intelligence, n.d.). Thus, the success or failure of Aml, will depend on the implications of issues such as energy, environment, social sustainability, privacy, social robustness and fault tolerance, as well as technological and economic feasibility. Regarding to privacy, studies have reported that, privacy protection is more important to the people than any potential benefits provided by technologies found in Aml applications. Therefore Aml systems should be tested to make sure they are safe to use. They should also be secure against deliberate misuse, to prevent it, Aml systems can assume security techniques like: ID authentication, micropayment systems and biometrics. Another solution is to obtain the minimal amount of personal information that it needs to achieve the user's goal (Commission, 2001; Cook et al., 2009).

According with (Commission, 2001), in the future the tangible and physical interface that divides people from Information and Communication Technologies (ICTs) should be imperceptible. In other words, Aml should be controllable by ordinary people, and the interaction with the system should be relaxing and enjoyable for the users, not requiring a perceptible learning effort to the users. This way there is a need to define human-centric computer interfaces that are context aware and natural. To emphasize the fact that devices must be imbued with an inherent consciousness about their current location and surrounding environment, this computing paradigm is also called sentient or context-aware computing. Creating systems that are very intuitive to use is a central challenge of Aml. Therefore, social and psychological aspects such as human factors must be taken in consideration during the design of intelligent interfaces. There are also supportive and technological developments user interfaces design that are multimodal (multi-user, multilingual, multi-channel and multipurpose) for speech, gesture, and patterns recognition. Developments in speech recognition will allow hands free interaction with personal ambient devices, this will have a big impact on the miniaturisation of devices (Commission, 2001; Intelligence, n.d.). Human-computer interaction was always an important area of study, and is a very important aspect of Aml. On one hand, there is a motivation to reduce the HCI. The system is supposed to use its intelligence to infer situations and user needs when it is needed, like it was previously said. This is the idea of an intelligent social user interface. On the other hand, a diversity of users may want to seek direct information with the system to indicate preferences, needs, among others. Aml systems should know when it is convenient to interrupt a user, and when to make a suggestion, but also when is more convenient to refrain from making a suggestion. The user can get tired of too much intervention from the system, and if the user decides not to pay attention anymore the system is useless (Cook et al., 2009).

2.2 Contributing Technologies

Ambient Intelligence refers to electronic solutions that allow environments to be mainly sensitive, adaptive and responsive to the presence of people. As result, technologies like: smart materials, microelectromechanical systems and sensor technologies, embedded systems, ubiquitous communications, I/O device technology, and adaptive software, are needed to make Aml a reality. Thus it can be concluded that Aml has a decisive relationship with several areas in computer science. ISTAG organized it in five key areas, which are: Sensing, reasoning, acting, human-centric computer interfaces and security, as its illustrated in the figure 1 (Commission, 2001). In the following topics it will be briefly described these five areas.

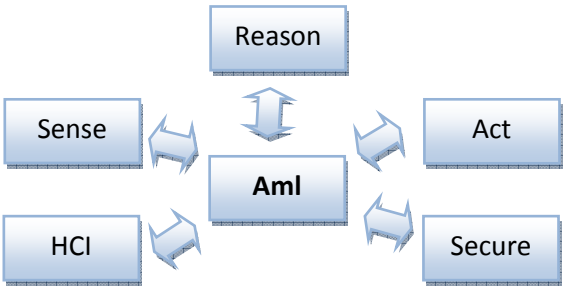

Figure 1 – Relationship between Aml and contributing technologies, adapted from (Cook et al., 2009).

2.2.1 Sensing

Aml is designed for real-world and physical environments, the sensing of the environment is accomplished through smart sensors. Sensors are physical components that allow an intelligent agent to sense and collect data from the environment, then uses this information to reason about the environment, and finally an action is taken in order to change the actual state of the environment. These devices which consist of sensing, are characterized by small size, low-cost compared to traditional sensors, low-power, have limited processing and computing resources, and have untethered communication in short distances. A large number of sensor nodes leverage the idea of a sensor network. These nodes are spread across a geographical area, and they collaborate amongst themselves with the intention to monitor physical or environmental conditions and communicate the retrieved data to the user. Usually the parameters that are monitored are: temperature, humidity, pressure, wind direction, speed, illumination intensity, vibration intensity, sound intensity, power-line voltage, chemical concentrations, pollutant levels, vital body functions, among others. Therefore and because sensors are a very unobtrusive hardware, they can be integrated into almost any Aml application (Cook et al., 2009).

Track and identify people in an environment is an important issue in Aml systems, because it allows the system to anticipate the needs of the user based on his preferences and behaviours, this way the system can serve the user more efficiently. Motion sensors are the technology often used for this purpose, but they only detect movement, they cannot distinguish the different users. To solve this problem it can be used Radio-Frequency Identification (RFID) tags, which consist in small devices with stored information to allow the identification of the person, animal, object to which it is attached. This technology responds to radio signals transmitted from a transmission base, some can even emit the sign itself. Another technology that can be used for tracking is the I-Buttons, which are small devices with a unique registration number. The I-Buttons can be used to identify persons or objects that place this device in the reader. To the I-Button be effective has to be physically placed over the reader, which makes it inappropriate to everyday use in a home (Cook et al., 2009).

Sensors generate large volumes of multidimensional data, which challenge conventional data analysis techniques. Some of the data can be noisy if the sensors are imprecise, or may be

missing values if a sensor fails. However, to analyse sensor data the Aml systems can use a centralized or distributed model. The sensors in the centralized model transmit data to a central server, which will analyse the received data. Whereas in a distributed model, each sensor performs local computation, and then communicates the partial results to the others nodes of the sensor network. So, the choice of the model has influences in the computational architecture, as well as, in the type of sensors that will be used. However, in this two approaches sensor data is collected from several sources and then combined to produce accurate information. The combination of information may be done by Kalman filters or probabilistic approaches. This process of filter and interpretation of the sensed data happens at the middleware level, which precedes the higher level decision-making modules. The scientific community has done a significant effort in achieving efficient and robust middleware levels, due to the importance that this has on maximizing the understanding of the environment through the sensed data (Cook et al., 2009).

2.2.2 Reasoning

Ambient intelligence applications have the capacity to reason about the collected data of the current environment, in order to know how to act in the current context. This reasoning is done through analysing the collected data by intelligent algorithms. To make this algorithms responsive and adaptive, it is used several types of reasoning, such as: modelling, activity prediction ad recognition, decision making, and spatial and temporal reasoning (Cook et al., 2009).

- **Modeling**

Modulation of the user behaviour is the feature that separates common computing algorithms from algorithms that are responsive to the user. A model can be used to customize the behaviour between the Aml system and the user, and if the model has an accurate enough baseline, it can be used to detect anomalies and changes in the activities of the users. Also, if a model can refine itself, then the environment can adapt itself to these changing patterns. According with (Cook et al., 2009), the Aml user modelling approaches are based in the following characteristics: the data used to build the model, the type of model that is built, and the nature of the model-building algorithm (supervised, unsupervised).

The data used to build the model is normally obtained from low-level sensors. This data is easy to collect and process, but the amount of data provided can be a computational challenge for modelling algorithms. For example, in the MavHome smart home project, which collects information about motion and lighting results in an average of 10,310 events by day. In order to deal with this amount of data, a data mining pre-processor search for common sequential patterns in the data, and then this patterns are used to create an a hierarchical model of users behaviour, called the Markov model.

- **Activity prediction and recognition**

Reasoning algorithms also have the ability to predict and recognize activities of the environment. For example, in a smart home environment the activities to be recognized can

be lifestyle patterns (such as, adequate food intake and sleeping). In a hospital environment, the activity to be recognized can be medicine intake (for example, ensure the right medicine is taken in the right quantity). Another example is in a smart car environment, which the activity to be recognized can be the driving behaviour (for example, to increase safety if driver is falling asleep). The MavHome project, among many others, adaptively controls home environments by predicting the location and activities of the users. In order, to anticipate the user's needs and assist when needed.

Researchers have explored a large number of approaches to recognize activities. These approaches vary according with the type of sensor data that is used for classification and the model that is used to learn activity definitions.

Regarding to sensor data, different types of sensor information are effective to classify different types of activities. For example, to recognize body motions (such as, running, sitting, among others) accelerometers located on the body can be used. To detect when a user is interacting with objects (such as keys, doors, among others) it can be installed state-change sensors on the objects, or it can be used RFID tags on the objects. This are just an example of the technologies that exist but many others can be used.

Regarding activity models, Nave Bayes has had promising results in activity recognition. Nave Bayes classifiers identify the activity, which corresponds with the highest probability of the set of sensor values that were observed. The classifiers assume that all the features are independent, but when there are large amounts of sample data, the classifiers have good accuracy despite this assumption. Others approaches to machine learning models for the recognition of activities can be used, like for example the decision trees. The decision trees generate rules that are easy and understandable by users, but has the disadvantage of being fragile with high precision numeric data. As an alternative to this it can be used the Markov models, dynamic Bayes networks and conditional random fields to encode the probabilistic sequence of sensor events (Cook et al., 2009).

- **Decision making**

Automated decision making and control techniques can be used to develop a fully automated Aml application, being this a difficult task just a few were already implemented (Cook et al., 2009). One of these applications was the Mozer's Adaptive Home (Mozer, 2004), which has the objective of determining the ideal settings for lights in the home, for that it uses a neural network and a learner. Another pioneering project is the iDorm project (Hagras et al., 2004), which implemented an automated living environment. A campus dorm environment is automated using fuzzy rules learned through the observation of the users behaviour. These fuzzy rules allow the environment to adapt to changing behaviour, because they can be added, modified and deleted. Unlike the previous project, the automation is based on imitating the users behaviour, which turns more difficult implementing alternative objectives, like for example energy efficiency.

- **Spatial and temporal reasoning**

To an Aml system make sensible decisions it is important to know where and when some events occurred. This information together with other, makes it possible to identify the type of activities being performed with more accuracy (Cook et al., 2009). For example, in an Aml environment that monitors activities in a home to prevent hazardous situations. If it is detected that someone turns on the cooker and leaves the kitchen over 10 minutes, then the system can take an action, like for example turning off the cooker automatically).

2.2.3 Acting

Aml systems can act in an independent way in order to iterate and affect the system users, through the use of intelligent and assistive devices. Another mechanism is through robots, which can provide a wide range of assistive tasks to support Aml. Research in robotics has progressed a lot over the years. For example, nowadays robot assistants can be found in nursing homes, which can have a considerate impact in people's daily life, especially in the elderly people. Another example is the museum traffic control project, where a robot generates cues in order to encourage the visitors to travel to places of the museum that normally are not visited (Cook et al., 2009).

2.2.4 HCI

A key aspect for the acceptance of Ambient intelligence highlighted by ISTAG is that Aml should be easy to live with (Commission, 2001). This way, it is important to define human-centric computer interfaces that are context aware and natural (Cook et al., 2009).

Context aware is the key to create Ambient intelligence applications. This computing paradigm emphasize the fact that devices must be conscious about their current location and surrounding environment (Cook et al., 2009). This way the system can provide more relevant services to the user, an example of context-aware service could be a planned route for a car drive or even a real-time traffic update. Context-awareness is afforded by wireless sensor networks, and refers to real-world characteristics, such as time, location, temperature, emotions, among many others (Sadri, 2011).

Another important aspect of Aml has to do with interaction. Over the last decades, efforts have been made to reduce the interaction between humans and computers. This means that the system should act independently, using its own intelligence to know what action should be taken and when. In other words, the system should act like a butler that observes the activities of the environment with the expectation of helping when needed. However, direct interaction with the system will still be needed in several cases. So, HCI always was and continues to be an important area of research. For the users the interaction with the system is fundamental, it does not matter the progress that has been made in the Aml field, if the technologies are difficult or unnatural for the users, they simply will not use it. This leads to that direct interaction with intelligent environments should now be replaced by more natural

interactions (Sadri, 2011). By natural interactions it means that the user should interact with the environment through common human communication capabilities and implicit actions. These natural interactions can be attainable by the following technologies: gesture recognition, motion tracking, facial expression recognition, speech processing, among others. Several interface mechanisms can be combined to form multi-modal interfaces (Cook et al., 2009). Below are briefly described some projects in Aml which are context aware and allow natural interactions.

The Cognitive Agent that Learns and Organizes (CALO) project is an AI-based personal assistant with the goals of being adaptive, unobtrusive and ubiquitous. The principal ability of CALO is to take autonomous control, it assists user actions and acts autonomously on behalf of the users. The CALO knowledge is acquired by its own through the use of multiple sensors and the analysis of information sources accessed by the user (such as email, web pages, etc.). This way, using the learned information CALO serves the user by executing tasks (such as, schedule tasks, track topics, summarize information to the user, among many other services) and anticipating future needs (Calo, n.d.). The driver's intent project of Massachusetts Institute of Technology (MIT) consists in an intelligent automobile co-pilot that watches all the actions of the driver and only interferes when needed (for example, if the driver is about to have an accident). This project uses a laser rangefinder to identify obstacles in the road, and an on-board camera, which together with facial expression recognition technology, detects for example if the driver falls asleep (Cook et al., 2009). Another example is the UCLA's HyperMedia Studio project, which automatically adjusts the lighting and the sound of a stage performance according with the artists' positions and movements (Mendelowitz & Burke, 2005).

2.2.5 Secure

The emergence of ambient intelligence brought many advantages to the humans, such as reducing the cognitive and/or physical effort that is required for the users to perform a task. However, Aml integrates several invisible computer technologies in people's everyday lives, which can provide issues like privacy and security (Sadri, 2011). This way, the acceptance of Aml will require an awareness of issues related to confidence and trust, it does not matter if great Aml applications are developed if the people will not use it in fear that their privacy will be compromised (Cook et al., 2009; Intelligence, n.d.). However, in some cases when the benefits of an Aml application reward these issues the users will accept it more easily. Several Aml researchers argue that the personalized Aml services provided to users can enable sharing of personal and private information with third parties. Privacy protection is more important to the majority of humans that any benefits provided by Aml applications. Also, what is considered invasion of privacy varies for different people, principally differs by age group and culture (Cook et al., 2009).

In terms of security and invasion of privacy the use of video cameras as a kind of sensor in Aml applications is a controversial area. This equipment can be very useful to collect large

amounts of information of the environment, but to the users having cameras monitoring all their movements and behaviours raises clear issues of privacy. However, not all applications that use video cameras will invade the privacy of its users, as well as neither all the applications that do not use video cameras to monitor will perform a better job of ensuing privacy.

Behind the loss of privacy, the sensor level, sensor reliability, handling error and installation errors also can provide security risks. The designer when implementing a sensor network must consider these factors, as well as, sensor communication channel reliability and security, and sensor data security. Encryption of the collected data can resolve some of the data privacy issues, but implement the necessary security using the minimal resources it is a major challenge (Cook et al., 2009).

In sum, privacy and security in Aml applications is a sensitive subject, since to Aml applications is essential collect large and detailed amounts of data about users' everyday activities over long periods of time, to provide personalized services which is one of the key objectives of Aml (Sadri, 2011). Realizing this without compromising the privacy and security of the data is a challenge that has evolved a great deal of research. Some projects of these researches focus on keeping the sensed data private, while others are using devices that act as secure keys to provide and receive information (Cook et al., 2009).

2.3 Application Areas

Nowadays there is a wide range of potential uses of an intelligent environment. Ambient Intelligences can greatly impact the society of today, it can automate aspects of the daily life, improve the use of resources (such as water, gas, electricity) and even increase the productivity at work. Below are summarized some Aml application areas and mentioned some existing implementations, the technologies that are needed for this implementations and the challenges that still exist. It is noteworthy to mention that not all of the applications described below incorporate the six features of Aml systems (sensitive, responsive, adaptive, transparent, ubiquitous, and intelligent), but they all reflect a subset of these features (Cook et al., 2009).

2.3.1 Smart homes

Smart homes are a good example of an Ambient Intelligence environment. Various objects in a house (such as home appliances, household items, temperature handling devices, among others) can incorporate sensors to collect information about their use, also some even can act without human intervention. An environment like this can ensure benefits such as: security, economy, comfort, among others. For example the safety can be increased by monitoring the activities in the house and providing assistance when a suspicious behaviour happens. An air conditioning adjusting the temperature automatically can be an example of comfort. Control

the use of the lights it may be beneficial economically. These are just several examples of the benefits that this technology can provide, many others exist.

Nowadays, there are several physical smart homes that were designed to be intelligent and to have the capacity to make decisions based on its actual state and on the behaviour of the house occupants. One example of this is the MavHome project at Washington State University and the University of Texas at Arlington. MavHome is an intelligent home environment, which perceives the environment through sensors and acts upon the environment through the use of actuators. In other words, the home collects information, reasons about it and acts in the environment accordingly. In order to meet the overall goals, such as minimizing the cost of maintaining the home and maximize the comfort of the residents. The activities in the house are observed to identify recurring patterns and compression-based predictors are employed to identify possible future activities. Thus, actions that the resident would typically perform manually (like for example, turn on the light when entering in a room), can be performed automatically by the environment. In one month the MavHomes reduced the daily interactions of the residents by 76% on average (Cook et al., 2009).

An example of intelligent use of energy is the ALADIN (Ambient Lighting Assistance for an Ageing Population) project, which has the goal of enhancing the overall physical and mental wellbeing of the older people and translate this into a low-cost open solution. The adaptive lighting can contribute considerably to a healthy sleep-wake cycle, which has a great impact especially on the comfort and wellbeing of the older people. This adaptive lighting system includes a smart open-loop control that can adapt several light parameters in response to the psycho-physiological information it receives (Cook et al., 2009).

Another ambient intelligence application is the Gator Tech Smart House project, which has as principal objective to assist older persons and individuals with special needs, in order to maximize their independence and wellbeing. This home incorporates a large number of sensors and actuators, which generate a large volume of data streams. This data streams are filtered through a package of OSGi (Open Services Gateway Initiative) services, providing opportunity for data folding, modelling, and encryption. Also, this project uses a self-sensing service that allows the remote monitoring of the older people that live in the house, as well as a way to intervene remotely if necessary (Cook et al., 2009).

The Georgia Institute of Technology developed the Aware Home project, which consists of two identical but independent living spaces, each one contains bedrooms, bathrooms, office, kitchen, dining and living rooms, there are only two shared areas: the basement and the control room where are computer services are centralized. This house incorporates ultrasonic sensors, Radio frequency (RF) technology and video, recognition through floor sensor and vision techniques, to track the human position. These tracking and sensing technologies can be used to find lost objects (such as keys, remote controls, wallets, among others). Each object that the user would like to track has a RF tag attached and a long-range indoor positioning system to track the object. When the user wants to track an object he interacts with the system through an LCD touch panel, and the system will guide the user to the missing

object using specialized audio cues, like for example “your wallet is in the car” (Cook et al., 2009).

Beyond these referred applications several companies have been developing smart homes over the past decades. For example, Siemens has been investing in some application areas, such as adaptive offices, intelligent cars and smart homes. Practically the smart homes Siemens proclaim ensure economy, entertainment and security. The user can interact with the system through touch screens, or remotely using for example a mobile phone. Also, Philips has been researching the HomeLab, which focuses on interaction and how the houses of today can impact on the wellbeing of their residents from three perspectives: Need to belong and share experiences, need for entertainment and relaxation and need to balance and organize people’s lives. The level of social awareness that has to be embedded in the Aml systems is an important feature that Philips try to ensure, to be adequate and acceptable to users (Cook et al., 2009).

2.3.2 Health monitoring and assistance

Ambient intelligence technologies can recognize activities, detect changes or anomalies, and even monitor diet and exercise. In order to help in the assistance of elderly and individuals with physical and cognitive disabilities, allowing them to remain in their homes as long as possible. Several Aml applications in the assistance area to elderly have been researched and developed over the last decades. One example is the Proactive Health Group created by Intel, which has the goal of researching and developing technologies to enhance the wellbeing of elderly. As the social network is an important aspect related with the quality of life of the older people, Intel developed systems that use wireless sensors to monitor the social life of an older person. The information collected about the social interactions that the individual had at home is provided to the caregivers, in order to advise them on how this aspect of a person’s life can be improved (Cook et al., 2009).

2.3.3 Hospitals

Nowadays there are many applications of Aml technologies in hospitals to increase the efficiency of the hospital services. Aml capabilities can be used to provide safety, for example, by only allowing authorized people to have access to some areas of the hospital. Also, it can be used to reduce the number of nurses needed in assisted care facilities, and to make them aware more quickly of the patients needs. In addition, tracking is typically used to locate nurses and doctors in the hospital and to warn them when they are urgently needed. Many others applications exist beyond these. For example, the Lutheran General Hospital of Chicago built the Yacktman Children’s CT Pavilion that uses Ambient Intelligence to entertain and help patients during their examination sessions. The system can tailor the lighting and wall/ceiling projections according with the patients matter of preference when they are in a particular room. The system identifies each patient by a RFID-encoded card. This projections are useful

to calm the anxiety of the patients or to guide them along the hospital, among others (Cook et al., 2009).

The Ulster Community Hospitals Trust of Northern Ireland has set up the PathFinder project in order to care for elderly and vulnerable people in their homes. For this purpose it was installed several sensors in 3000 homes of the community, to monitor the elderly wellbeing. This way, the level of autonomy, independence and safety of the older people can be increased, especially if they have a medical condition that may be detrimental to their quality of life (Cook et al., 2009).

2.3.4 Transportation

Transportation facilities such as cars, buses, even train stations, among others, can be equipped with Aml technology to provide fundamental knowledge about the system performance in each moment. This Aml technology can be GPS-based spatial location, vehicle identification and image processing to make the transport more fluent, efficient and safe (Cook et al., 2009). This way, preventive actions can be applied based on this knowledge, ensuring the experience of people using that transport.

For example, the I-VAITs project has the objective of providing assistance to drivers by collecting important information, such as: the way they use different elements of the car, the movements and facial expressions that they make, among others. Thus, the system can assist the driver when he needs, like for example when he is executing tricky manoeuvres. Similarly, Pentland in partnership with Nissan Cambridge Basic Research, has developed a system that allows the car to observe the driver, by continuously estimating the internal state of the driver and responding appropriately. It was created an Hidden Markov Model (HMM) of the driver's hand and leg motions and associated actions, such as passing, stopping, turning, speeding up, with the objective of classifying a real driver's actions in relation to the artificial model. The system could identify which action the driver was performing as soon as the action started with high accuracy (97% within 0.5s of the beginning of an action, rising to over 99% accuracy within two seconds) (Cook et al., 2009). This quick driver's actions detection allows real-time optimization of the car's performance to alert the driver of potential dangers that may exist in any given situation.

2.3.5 Emergency Services

In fire brigades, the emergency services can quickly and efficiently locate and chart a route to a place where a hazard is occurring or is likely to occur, improving thus the response to that hazard. This can be done through image processing and traffic monitoring like in the e-Road project. Alike, the Prismatica project uses cameras that monitor public transportation locations in order to detect situations such as: overcrowding, the presence of people and objects that are not moving, motion in a forbidden direction and intrusion (Cook et al., 2009).

Thus the environment and officials can respond more quickly and more efficient to the suspicious behaviour, ensuring a higher level of security.

2.3.6 Education

The schools can use Aml technology to track the students' progression on their studies, in order to improve the learning experience. For example, the smart classroom of Shi, et al., provides devices such as an interactive whiteboard that allows people to write notes on the board using a digital pen. The classroom is also equipped with video and microphones that recognize a set of gestures, motions and speech that can be used to retrieve information or to capture the students' attention to the subject being taught. At the Arizona State University, the Reconfigurable Context Sensitive Middleware (RCSM) Project has the objective of improving the collaborative learning in the classrooms. RCSM allows the monitoring of noise, light, mobility activity, and allows automatic distribution of the presentation materials, as well as group discussion between students and the professor, among others. Another Aml application in the education area is the Smart Kindergarten project at UCLA, where a Smart Table was developed to monitor the kids' interaction with blocks on the table surface, this way the teachers can observe the learning progress of the children (Cook et al., 2009).

2.3.7 Workplaces

Ambient intelligence can be very useful in a workplace environment, it helps workers focus on their jobs without having to deal with complicated technology. One example of that is the Interactive Room (iRoom) project at Stanford, which provides an easy retrieval and display of useful information. For example, the users can display URLs (Uniform Resource Locator) on a surface just by dragging the URL onto the appropriate PDA icon. Another example is the MOSES system that uses Aml to track the workers location and to monitor what tasks they are performing. The system uses RFID technology to recognize positioning of the important elements that form part of the environment. For that, the employees need to be equipped with RFID readers so that the system can track them, the system also can remember the worker what tasks still need to be done (Cook et al., 2009).

2.4 Chapter Summary

In this chapter it was presented the ambient intelligence definition in a general context. Like it was referred, Aml is an emerging paradigm in information technology, in which people are surrounded by a digital environment that is aware of their presence and actions, and is responsive to their needs. As could be seen in the section 2.2 Aml is related with several areas in computer science, such as: sensing, reasoning, acting, human-centric computer interfaces (HCI) and security. These five areas enhance the key factor of Aml applications, which is the presence of intelligence. This means that the Aml algorithm observes the state of the

environment through sensors, reasons about the collected data of the environment using AI techniques, and then acts upon the environment in order to achieve its goals. Beyond that, the HCI and security areas are important for the acceptance of Aml applications by its users. To Aml applications be widely accepted and useful to the society, improvements are needed in several levels: infrastructure, algorithms and human-computer interaction (Cook et al., 2009). If the users find out these applications are too intrusive, difficult to use or even that compromise their privacy, they simply will not use it. Therefore, during the design of intelligent interfaces it is important to take in consideration that Aml applications should be driven by social and psychological aspects, such as human factors (Commission, 2001; Intelligence, n.d.).

Also, in the section 2.3 were highlighted several application areas of Aml, such as smart homes, health monitoring and assistance, hospitals, transportation, emergency services, education, workplaces. In these, it was stated where Aml technologies can be used and for what purpose, as well was referred some of the Aml projects that already were studied, developed and implemented by researchers.

In sum, Ambient intelligence offers many potential advantages to users like: customizing their environments and unobtrusively meeting their needs, supplying users with customized information reducing the cognitive or physical effort that is required to perform a task (Cook et al., 2009). However others researches and provisions will be necessary to make sure that it has the maximum positive effect on the personal life of the users and society (Commission, 2001).

3 Human-Computer Interaction

The methods by which humans have been interacting with computers have been an important field of research in the last few decades. In this chapter the area of study denominated Human-Computer Interaction (HCI) is approached. The HCI field involves the study, planning and design of how people and computers interact with each other, in order to satisfy in the most effective way the user's needs (Galitz, 2007).

The next topics intend to provide an overview on the state of the art of HCI systems and cover the most important branches of this subject. The definition and characterization of a user interface will be presented in order to understand its importance and relevance in an interactive computational system. Section 3.2 presents an overview of the history of human-computer interface and addresses the recent technology advances in the field. Section 3.3 circumscribes the different parts of user interface development, as well as the importance of the usability of an interactive system, taking into account the user role along the software development process. Finally, in section 3.4 two methodologies of user interaction development will be presented and characterized: star life cycle and useware engineering. These methodologies are user-centred, in order to ensure usability of the final system.

3.1 Definition and Terminology of Human-Computer Interaction

The concept of human-computer interaction exists since the emergence of the first computers (Hix & Hartson, 1993). The research in this area has been growing fast in the last decades, new designs of technologies and systems appear every day. Thus the interaction between the user and the system is increasingly an aspect to take into account when developing the system. The reason is simple, it does not matter how sophisticated a computer system is, if it is

difficult or unnatural to use for the users, they simply will not use it again. Two main terms should be considered during the design of HCI: functionality and usability (Karray et al., 2008). Functionality of a system can be defined by what the system can do, in other words, the set of actions and services that system provides in order to accomplish the goals of the system. However, the value of functionality is only visible if the user uses the system efficiently. Usability of a system is related with how well it can be used efficiently and adequately to achieve the purpose of the system. The effectiveness of a system is achieved when there is a balance between the usability and functionality of the system (Karray et al., 2008).

The study of human-computer interaction has the goal of explaining how to provide the best interaction, this means, provide a high degree of usability (Hix & Hartson, 1993). Know what makes a certain HCI design good is not easy, it is mostly subjective and context dependent (Karray et al., 2008). HCI designers when designing a user interface must have special attention to the characteristics, goals and context of the individual user interface, as well as, the needs and desires of the users. The interface represents the state and behaviour of a computational system, allowing the user to control or perform tasks. It can be defined as a conceptual borderline between the user and the computational system, with the goal of managing the process of communication between them (Hix & Hartson, 1993). In the next topic, the issue of why the user interface is so important is addressed, and how it can effectively affect the acceptance of the system.

3.1.1 The Importance of the User Interface

Over the past few decades, user interfaces have gained greater importance in relation to what they used to have. The evolution of personal computers and their lower prices made them available to more people and used for a wide variety of different tasks. Also, users have become increasingly demanding with the quality of the interfaces, since the appearance of certain software showed that it is possible to have a pleasant and accessible interface (Nielsen, 1993).

User interface design is a subset of the field of human-computer interaction. It is the most important part of any interactive computer system, because it is the window to view the capabilities of the system. For many users the system is only the interface, because it is the only component that they can see. Nowadays, over half of programming code produced for an interactive system is to support the user interface. In some cases, the amount of programming code devoted to the user interface is close to 100% (Galitz, 2007; Hix & Hartson, 1993). Comparing the user interfaces of today with the user interfaces of the past, it can be concluded that the design of interfaces had a great progress. However, in these days there still are several bad cases of interface design, much still remains to be improved (Galitz, 2007).

In the design of a user interface, HCI designers must always consider: what tasks the user wants the system to execute; how users should work and organize their tasks; which limitations users may have; what users enjoy in a software; and beyond that they must consider the limitations of the computer hardware and software (Galitz, 2007). The process of

the user interface design is not easy and it is not common sense. Just because anyone can design a user interface it does not mean that it will be an usable interface (Hix & Hartson, 1993; Hussmann, Meixner, & Zuehlke, 2011). HCI designers have as major goal to provide harmony between the user and the computer. For that they must give attention to the user interface, and know the user (Hix & Hartson, 1993). The user has an important role in the interface development process. This user role will be described with more detail along the next topics.

The user interface is composed by two components, which are input and output. The input component is how the user communicates with the computer, which can be done via keyboard, mouse, trackball, touch-sensitive screens or voice systems. The output is how the computer conveys the result of the actions requested by the user. The output mechanisms used today are the display screen, voice and sound (which are an advantage for users with visual impairment). Output mechanisms similar with human senses like smell and touch, still remain largely unexplored. A well-designed interface design must provide a mix of well-designed input and output mechanisms, thus satisfying in an effective way the needs and limitations of the users. Moreover, a good user interface should allow the user to concentrate on the task being performed, rather than focusing on the mechanisms that are being used to perform the task (Galitz, 2007).

The screens layout and appearance also can make all the difference in the human-computer interaction process. If they are inefficient and confusing, the users productivity can be affected, because they will have difficulties doing their job, and will spend extra time trying to understand the system and correcting their mistakes. In addition, a bad interface can affect the acceptance of the product, because it can cause frustration and increase stress in the users, which may cause some users to abandon the software permanently. Studies have shown the benefits of a well-designed interface. According with (Galitz, 2007), a researcher tried to improve an interface, making it clearer and more noticeable. Independent items that were combined on the same display to conserve space were separated and organized in the interface, in order to increase the readability. The results showed that the users were about 20 percent more productive, and the number of mistakes also decreased with that version of the interface. Thus, it can be concluded that a proper formatting of the presented information in the interfaces could have a significant positive effect in the users' performance. This performance improvement can result in a reduction in the cost of personware to the organizations. Presently, the cost of personware is an important fact in purchasing an interactive system. The cost of personware includes training and daily usage costs for users (Hix & Hartson, 1993). In addition, if an interface is well-designed, the company can save in the users training, because if the system is clearer and easy to learn the training time is significantly reduced, and support line costs are lowered because less assist calls are necessary (Galitz, 2007). In conclusion, a well-designed interface can make the organization save large amounts of money, but unfortunately sometimes these savings are not immediately visible to the organization, and may simply appear after the product launch (Nielsen, 1993). Furthermore, it should be noted that the user interfaces are a good way to highlight and differentiate systems. Since the current market brings increasingly technically

and functionally similar products, the usability can be used as an additional sale argument (Hussmann et al., 2011; Nielsen, 1993).

3.2 A Brief History of the Human-Computer Interface

Nowadays, the human-computer interaction has become an important area of study, because for the user the interaction with the system is at least as important as computation of the system (Hix & Hartson, 1993).

In the history of computing, it is possible to observe considerable changes in the development of hardware parallel to that in the user interface technology. Computer designers have been developing human-computer interaction methods that are easier to learn and use, like the systems that recognize human speech. This type of interaction can also help with the deficiencies and limitations that users may have (Galitz, 2007). The categories of people using computers have changed. In the past, computers were used especially in a professional environment, and nowadays they are used by everyone (Nielsen, 1993). In conclusion, the research done over the last decades, in Human-Computer Interaction has been very successful and has fundamentally changed computing (Galitz, 2007). Below, is shown a brief summary of user interfaces generations.

3.2.1 Batch Systems

Batch systems were the first generation of user interfaces and are not a highly usable interaction. They used to automate the sequence of operations involving the execution of a program. The user joins the program to the set of existing programs, creating a batch and each batch is sequentially executed by the computer. The results are supplied to the user, as the batch processing ends. The user interaction with the system when the batches are running is not possible, so all the user's commands had to be specified before the results or its execution. So as previously stated, this type of system was not highly usable, because if the user made a mistake entering the last command, it is necessary to wait for the batch computation to finish, which was very frustrating (Nielsen, 1993).

3.2.2 Line-Oriented Interfaces

Line-oriented interfaces were known as one-dimensional interfaces, where the user could only interact with the system through a command line. This way, the user could only edit one line of text at a time, instead of moving around in the document as in screen-oriented editors or graphical user interfaces. The user could edit an individual line by inserting and removing characters and moving the cursor left and right. After the user edited one line and hit the return key to the next line, the input data could not be modified anymore. Similarly, after the system presented a line of information to the user, it also could not be modified any further. The system did not have the capacity of updating the data already in the screen, in order to

reflect any changes that may have occurred in the data. Since users could not move within the screen, their interaction with the system was limited to question-answer dialogues and by typing commands with parameters. In question-answer dialogs the computer request answers to the user, one at a time. But it had the problem, that the user could not change a previous answer and had to reply to the current question without knowing the following questions (Nielsen, 1993). However, question-answer dialogs can be suitable in situations where the dialog is well structured (so that the user does not feel the necessity of changing a previous response), and where it is acceptable the user being directed by the system rather than performing the task in alternative ways.

Many line-oriented user interfaces were built on command languages, which are very powerful and allow the construction of very complex sequences of commands using sets of modifiers and parameters. But, command languages had the disadvantage of being more prone to errors, because they required the user to specify the desired commands exactly in the same required format. The user had to remember the command and parameters syntax without the computer help. Most command languages allow the user to abbreviate the command names, in order to reduce this problem and speed up the interactions (Nielsen, 1993).

3.2.3 Full-Screen Interfaces

Full-screen interfaces are two-dimensional interfaces and opposite of the line-oriented interfaces. This kind of interfaces allows the screen output to be modified. Form-filling dialogues are a classic use of the full-screen interfaces. The system presents to the user a group of various labelled fields that can be edited in any sequence desired by the user. In many systems with full-screen interfaces the primary interaction style is made via function keys. These function keys have the advantage of serving as interaction accelerators. The exact interpretation of a function key may depend of the screen object pointed by the cursor, thus making the use of function keys an early approximation of the point-and-click interaction style of modern mouse-based interfaces (Nielsen, 1993).

3.2.4 Graphical User Interfaces

Graphical user interfaces, also called WIMP systems (windows, icons, menus, pointing device), are the most common user interfaces that exist nowadays. This type of interfaces are based on visual representations of the dialogue objects, which allow the user to control the dialogue by moving and manipulating objects around the screen, with an input dispositive such as the mouse. Graphical user interfaces are almost three-dimensional, because they allow the overlapping of windows. But in reality, overlapping windows is not a truly three-dimensional experience, because it is not possible to see the content of a window behind another, without moving the windows. So it is more correct to refer to these interfaces as having two-and-a-half dimensions.

In this type of interfaces, the primary interaction style is direct manipulation, where the user performs a task by selecting an object and then performing an action on that object (e.g. the user selects an icon, and then deletes it) (Hix & Hartson, 1993; Nielsen, 1993). The use of a pointing device such as the mouse, offers the user a sense of control over the system and for the user it is a natural way to interact with the interface, which is more appealing and noticeable due to the use of graphic design (Nielsen, 1993).

3.2.5 Next-Generation Interfaces

The research done over the last decades in Human-Computer Interaction has been very successful, and has fundamentally changed computing (Galitz, 2007). In the next generations of user interfaces it is expected that the number of dimensions of a user interface continues to increase. Some ways to increase dimensionally a user interface are for example: sound and voice, as well as a true third spatial dimension in the form of virtual reality systems. Another prediction regarding the next generation interfaces is that they will be more object-oriented in terms of their functionality, and not just in terms of information manipulation.

The principle of an explicit interaction between the user and the system in which the user orders the system to do a certain action is present in all the user interface styles referred above. But it is expected that in the next generation interfaces, it will be the system taking over responsibility for the interaction, basing its actions on its observations of the user. This way, the user will focus on the environment and on the task being performed, instead of having to deal with controlling the system explicitly. For example, if the computer detects that the user is distant from the display, the system can enlarge the text on the display, in order to be readable for the user. These future systems can be achieved through technologies like: eyetracking, gesture recognition and semi-intelligent analyses of the user's actions (Nielsen, 1993).

3.3 User Interface Development

User interface development is different from the Interaction development. As it can be seen in the figure 2, user interface development is composed by two parts: development of the interaction component and development of the interface software. The interaction component defines the behaviour of the interface in response to the user actions, while the interface software component supports the interaction component. However, in practice this separation does not always happen, the majority of the software engineers were responsible for the development of the two components. As a result, it were developed user interfaces with quality and usability variable (Hix & Hartson, 1993).

Development of the user interface	
Development of the interaction component	Development of the interface software

Figure 2 – The parts of user interface development, adapted from (Hix & Hartson, 1993).

Due to the difference between these two components, the development occurs in different domains (behavioural domain and constructional domain). These different domains correspond respectively to the working groups of the professionals that design and develop the interaction component and the professionals that design and develop the user interface software.

In the behavioural domain, the interaction between the user and the interface is described abstractly, this means, it is described independently from the software. This domain involves human factors, guidelines, human cognitive limitations, graphic design, interaction styles, usability specifications, rapid prototyping and formative evaluation, among others. In the constructional domain the software developers develop the software that implements the behavioural design. This domain involves algorithms, programming, widgets, procedure libraries, control and data flow, state transition diagrams, callbacks, among others (Hix & Hartson, 1993).

According to (Hix & Hartson, 1993), approaching user interface development in the behavioural domain (from a user and task view) increases the usability instead of approaching it from the constructional domain (from the system view). Both domains are necessary, the merged results of both domains determine the usability of the final product. The majority of the usability problems are a result from a conflict of interests between the programmer and the user: “What is best for a user is rarely easiest for a programmer”.

3.3.1 Usability in Human-Computer Interaction

Usability has become a key issue (Hix & Hartson, 1993; Nielsen, 1993). However, it was not always like that, sometimes software engineers and managers see the user interface as a minor problem, something that does not interfere with the sale or with the acceptance of the product. Studies have shown that a highly usable interface can bring many advantages for both users’ and the company. Because, an interface with usability problems can make users waste their time trying to understand the interface or trying to recover from their mistakes, this can result in additional cost to the organization in both labour costs and lost sales opportunities (Nielsen, 1993).

Usability is a quality attribute that assesses how easy user interfaces are to use, and is related to the effectiveness and efficiency of the user interface and to the potential to fulfil the users goals, providing them satisfaction (Hix & Hartson, 1993). Usability has multiple components and could be defined by the next five usability attributes (Nielsen, 1993):

- **Learnability:** The system should be easy to use, even for users that will use it for the first time;
- **Efficiency:** After the users have learned the design, they must be able to perform tasks quickly, thus a high level of productivity will be possible;
- **Memorability:** When a user returns to the system after being a long period of time without using it, they should remember how it works, without having to learn everything all over again;
- **Errors:** The system should always have a low error rate, when users make mistakes they should easily and rapidly recover from them, and catastrophic errors must never occur;
- **Satisfaction:** The system design should be pleasant to use for users.

The most important aspect of usability is to know the user, in other words, know the users' task and users' individual characteristics and differences. The users can be considered from novice to experts, according to their level of experience with the interface. Some interfaces will only be used by novice users; for example, a kiosk for making dinner reservations in an amusement park or an installation program. These interfaces will be used for the same user once or just a few times, so they should be as simple and usable as possible. But most often, an interface will be used by both novice and expert users, but even expert users are not experts in all the system, because some systems are so wide and complex that any given user will only use a small subset. Therefore the interface needs to be prepared for both types of users. For example, an interface can include accelerators to allow the expert users to do their tasks faster and it can also include online help to assist the novice users. The users individual characteristics and differences like age, gender, spatial memory, reasoning abilities and learning style, also have an impact on the interaction between human and computer. In conclusion, in the design of a user interface it is necessary to consider all kinds of intended users, ensuring that the interface is used by as many as possible. Beyond that, knowing the user can avoid wasted time and development efforts, such as developing features that the users do not want or need (Nielsen, 1993).

Usability can be measured by having a number of test users using the system. In this test the users should be the end users or should be representatives as possible of the end users. An important aspect in measuring usability is that it is made relative to certain users and certain tasks, because it can happen that the same system can be measured as having different usability characteristics if tested by different users for different tasks (Nielsen, 1993).

It is important to note that the optimal user interaction concept is impossible to determine, because usability is a subjectively experienced non-functional requirement, depending on several factors characteristic of users such as skills and experience, so each user will make their own interpretation of the same interface (Hussmann et al., 2011; Nielsen, 1993). The best way to make the best user interface design possible is to understand the users and their tasks. However, sometimes the users do not know what is best for them, and sometimes they

have different opinions when asked about the details of the user interface design. Thus, the interface designers should not only follow the opinions of the users (Nielsen, 1993).

3.3.2 Human Factors

To develop an interactive system that is easy to learn and use, it is necessary to take into account the user behaviour, in other words, the human factors. The study of human factors includes the derivation of principles of human behaviour, across empirical testing using human participants. The goal is to optimize human performance, including error reduction, increased throughput and increased user comfort and satisfaction. Empirical testing allows the measuring of the human performance using an interactive system and it involves several phases, including formation of a hypothesis, design of a study with participants' representative of end users', collection of performance data through observation of these users performing certain tasks, data analysis, and confirmation or refutation of the hypothesis (Hix & Hartson, 1993).

The study of human factors allows the interaction designers understand what they need to do, so that the interface is natural and intuitive to the user, because it is the interface that must be adapted to the user and not vice versa. It is worth noting that a perfect guidance in interaction design ensuring usability in interfaces does not exist, each situation is different (Hix & Hartson, 1993). Below are referred four kinds of human factors information that could influence the product:

- **User interaction standards**

  User interaction standards are official and publicly available documents, which provide requirements for user interaction design. They must be followed during the design of the user interaction, and are enforceable by contract or law. Standards are extremely difficult to interpret and enforce, because they have very general wording that it is often difficult to determine whether they are being followed or not. Standards require much interpretation and tailoring so that they can be useful in user interaction design. Their major advantage is that they draw attention to the user interface, but they are too vague to offer effective guidance (Hix & Hartson, 1993).

- **User interaction design guidelines**

  Guidelines are published in books, reports and articles that are publicly available. They are not specific to a single organization, but rather apply across the broad spectrum of user interaction design. They offer flexible guides to define a common look and feel of the user interface, but they must be customized in order to produce specific design rules. These guides are considered common sense, but in reality guidelines are not just common sense. Because they are general in their applicability, which can often make them contradictory, thus it is necessary a deeper experience and expert knowledge in the user interaction design to know which and how to apply in a particular design. The main difference between guidelines and standards is that guidelines serve more as suggestions on how to

produce a good interface, whereas standards are enforceable in a user interface and defined by contract or by law (Hix & Hartson, 1993).

- **Commercial style guides**

Commercial style guides are typically produced and commercialized by an organization or a vendor. A style guide document provides concrete and useful framework for design than does a standards document. This document typically includes a description of a specific interaction style or object, including both its look (appearance) and feel (behaviour), and includes guidance on when and how to use a particular interaction style or object. Also, the commercial style guides can contain descriptions of particular interaction components, like for example windows, menus, dialogue boxes and controls, keyboard and mouse usage, and even description of messages and help design. To better show how sample screens should look, besides the textual descriptions, the style guide document can contain sketches of examples of the interaction components.

Many commercial style guides are associated with a toolkit that supports the style defined by the corresponding guide. The toolkits contain precoded interaction objects that can be used in the implementation of a user interface. The related style guide describes each interaction object's default appearance and behaviour. To allow customization of the objects, many toolkits support changes to the precoded objects.

There are some aspects to consider about style guides. Firstly, many commercial style guides have not had a great deal of professional human factors input or evaluation. But as they have reasonable usability, they provide a decent starting point for objects to be used in an interaction design. Also, it is noteworthy that a style guide is not sufficient to ensure usability of an interface, because a style guide can remove some of the constant decision making inherent in interaction objects and styles. Lastly, a style guide tends to provide specific guidance on what a particular interaction object should look like and how it should behave. However, they do not always inform about when a particular interaction object should be used in the interface, which is one of the hardest design decisions.

The main advantage of commercial style guides is to improve consistency of the user interaction design. They are very detailed as to the appearance and behaviour of the interaction objects, and if the style guide document is well produced it does not require too much interpretation, unlike standards and guidelines (Hix & Hartson, 1993).

- **Customized style guides**

In a user interface development, it is important that the development team works together and focus on human factors issues to produce a document with their own customized style guide. This document can be produced for one particular interface development project or for multiple projects, and it is internal to an organization. The customized style guide should contain very detailed recommendations of various aspects of the user interaction design. The main advantage is that it provides consistent and explicit information for a design. When the customized style guide is concluded, the

developers can use it as a record of decisions that have been made about a user interaction design, this way they do not have to constantly remake these decisions (Hix & Hartson, 1993).

The main difference between guidelines and commercial style guides, as compared with customized style guides, is that guidelines and commercial guides are applicable across a broad range of user interfaces, while customized style guides are normally applicable for a specific project, product, and/or organization. Customized style guide documents should contain the following information (Hix & Hartson, 1993):

- An introduction that should explain the basic user interface paradigm used in the guide;
- A section where it is indicated the input and output devices for which the guide is appropriate, and an explanation of the intended use of these devices in an interface;
- A sketch of a screen template and a description of a typical screen format and layout, in as much detail as possible;
- A section for each interaction object and interaction style that is included in the guide, for each style it should be described what it is, what it looks like, how the user interacts with it and when it should be used in the interface;
- A section about messages (like error messages, informative messages, warning messages, among others).

Finally, it is noteworthy that to produce a really effective customized style guide, it is better to be done by a human factors expert early in the development process. The effort and cost to produce this guide will quickly be amortized, because the developers do not have to remake design decisions constantly, like it was mentioned above.

3.4 User Interaction Development Process

Many of the software developed in the past used a top-down approach based on functional decomposition. The life cycle associated with this kind of methodology is called the waterfall method, because it follows sequentially from one phase to another as shown in the figure 3.

Top-down development often begins in formalizing specifications, which helps software developers have a better understanding of the problem. In many development environments, the process of software development is correctness-driven, this means that the implementation should follow and be faithful to the specifications. As seen in the figure above, the process is strictly sequential, but still exists testing at some points of the sequence, and each activity has a feedback path to the previous activity.

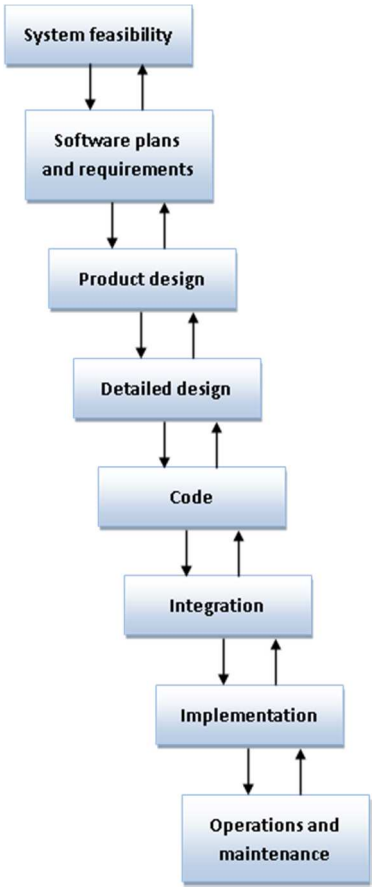

Figure 3 – The waterfall method for software development, adapted from (Hix & Hartson, 1993).

According with the study of (Hix & Hartson, 1993), a life cycle would benefit from a less rigid sequential process, because in a rigid sequential process (top-down development) the software cannot be tested until late stages of the development process, and if problems are not detected early, they can be costly to remedy later. Observing interaction developers work, can be seen that they mostly work in alternating waves of two complementary kinds of activities (top-down and bottom-up), because it is easier to learn by starting with concrete examples that work toward the abstract. In the top-down model, high-level requirements are documented first, and then the next step is design and build. In the bottom-up model, the lowest level of functionality are designed and programmed first, and finally all the pieces are integrated together into the final software. The top-down model tends to reflect a system view, working toward the user, and the bottom-up model tends to reflect the user's view, working toward the system. This kind of work goes against the traditional rigid sequential process, but it is how most developers work in reality. Also, it is a good way for rapid prototyping. This means that there are developed early versions of the system that illustrates essential features of the final product. There is something to evaluate much sooner in the development life cycle, which brings the advantage of making substantial changes if needed to the product. Thus, there are multiple iterations through the refinement process and a better understanding of the needs of the user, improving the usability of the final system (Hix & Hartson, 1993).

Generally, it is not feasible to develop a complex system sequentially from start to the end. The user interface development could benefit from the use of an iterative methodology (Hix & Hartson, 1993). In an interactive system, the user is the element that has unpredictable behaviour. The interaction development should be essential and inherently iterative, so it can be a process of self-correction. Next, it is addressed two methodologies used in the development of user interfaces (star life cycle and useware engineering).

3.4.1 The Star Life Cycle for User Interaction Development

Due to the unpredictably human behaviour, changes in the specifications will likely exist along the software development process, because there are always going to exist improvements to be achieved, mainly after the interaction developers observe the user interacting with the system. All this considerations led (Hix & Hartson, 1993) to propose a new life cycle for the development of interactive systems, the Star Life Cycle. This methodology is an iterative process and minimizes the number of ordering constraints among the development activities, allowing the alternation between top-down and bottom-up development in an environment of constant evaluation and iteration (Hix & Hartson, 1993). Thus, the Star Life Cycle is especially suited for the needs of user interaction development.

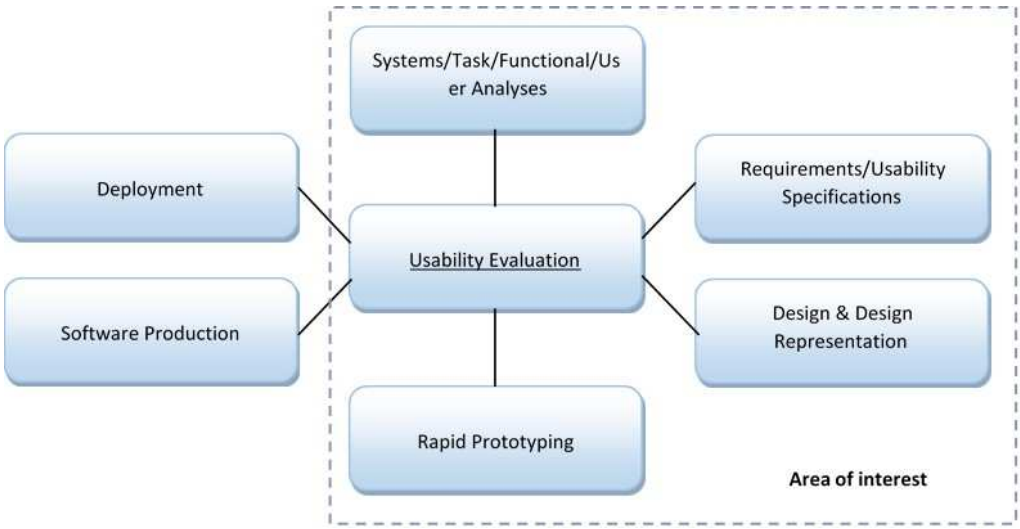

Figure 4 – The star life cycle for user interaction development, adapted from (Hix & Hartson, 1993).

This methodology is called star life cycle because of its shape, where each point of the star is one activity and these activities are not ordered or connected in a sequence. This implies that the development can be started in any activity, and is not necessary to follow any determinate sequence of activities. For example, the user interface developer does not need to specify all requirements before start working on design, he can start prototyping and during the process improve the requirements. As can be seen in the figure above, all the activities in the star life cycle are highly interconnected, but the usability evaluation process is the centre activity. This means that the life cycle is evaluation-centred, so, each activity is evaluated before going to the next activity. Another aspect of this methodology is the fact that the end of the

development process is not clearly identified. So, the user interaction development process in this methodology is concluded when the criteria defined in the usability evaluation phase are achieved. The five activities addressed in the area of interest, identified in the figure above, are related with the development and evaluation of an interface prototype (Hix & Hartson, 1993):

- **System/Task/Function/User analyses**

In this activity should be analysed the purpose of the new system to be developed, as well as, the market requirements and organization goals. Thus it can be predicted if the system will be successful. The tasks that the users and the system should perform together are also analysed and identified, as well the necessary resources to the realization of these tasks. Also, should be analysed the functionalities that the system should perform. Decisions about what should be performed by the user and what should be performed by the system should also be performed in this activity. A study about the final users of the system should be made, in order to classify the users based on the individual characteristics, needs and expectations. It is created a set of users profiles, which can contribute to increase the usability of the system.

- **Requirements/usability specifications**

The specification of usability requirements is a central activity in the development of user interfaces. The definition of quantitative usability goals can be used as a guide to know when a particular interface is good enough. The user-centred evaluation approach ensures that changes between interactions are made. However, there are no guarantees that these changes will increase the interface usability. So, the specifications of usability requirements will allow assess the usability of a user interface in each interaction.

- **Design & design representation**

The design and the design representation are two different fields. The design is a creative mental process to the problems resolution. The design representation is the physical process of design storage.

One of the factors that influence the quality of the interaction design is the capacity of the people involved in the system development in understand the design (Hix & Hartson, 1993). This is even more important in interactive development methodologies, where there are several individuals that participate in the development process with distinct roles but cooperating, such as designer, programmer, expert, appraiser, documentation specialist, amongst others. Each one of these stakeholders has specific communication needs. To the design be completed understood by all individuals involved in the development process the use of representation techniques is necessary.

Techniques that describe the interaction from the point of view of the system (like for example state diagram) are from the constructive domain, this means that only represent the software component, not the interaction part. While, techniques from the behavioural domain are user centred, task oriented, and influenced by the user requisites. The behavioural design must be translated into constructive design, in order to represent the system vision of

how the behaviour is supported. One of the techniques of the behavioural representation is the User Action Notation (UAN), which was created within the Human-Computer Interaction Project at Virginia Tech. The UAN is user and task oriented notation, which describes the behaviour of the user and interface when they perform a task together. The interface is represented by a hierarchical structure of asynchronous tasks. The user actions, the system and the information about the change of state are represented at a lower level. At all levels, the users' actions and the tasks are combined, based on temporal relations, in order to describe the allowed behaviour. Also, the UAN notation can be complemented with scenarios, state diagrams (shows the interface state changes as a result of the actions taken by the user), and with documentation about design decisions.

- **Rapid prototyping**

In the life cycle methodology, the users have an important role during the overall development process. They accompany the progress of the interface giving feedback to the developers. In rapid prototyping, the process of building a prototype is accelerated, so that the time between the prototype construction and respective testing by potential users is sufficiently short to allow substantial changes if needed. So, the development process could have many interactions. Each interaction refines the interface through the feedback of the users, so that usability of the system increases.

Like it was previously referred, the Star Life Cycle is evaluation-centred, so this iterative development process is dependent on the existence of prototypes to evaluate the interaction design and to make changes to the course of the development process based on that assessment, until the criteria defined in the usability evaluation phase are achieved.

- **Usability evaluation**

This activity is the central node of the star methodology, and consists in evaluating if the interface is successfully achieved according with the usability requirements specified in the requirements/usability specifications activity. The interaction design is evaluated at the same time that is developed, since its beginning and continuously along the development process interface. Thus, this type of assessment is called formative evaluation. Where an evaluation session takes place and a group of participants representative of the final users perform a usability test. In this test it is observed the users interacting with the prototype, in order to perform a set of tasks asked to them. During and after this evaluation session data is collected, such as: the performance and the opinions of the participants. The analyses of this data will allow developers to known what they should change in the prototype development, in order to try and ensure the system usability. After the developers implement the new changes and features of the prototype, a new evaluation session will be performed.

In the formative evaluation are collected different types of data during the prototype test: objective and subjective; qualitative and quantitative (Hix & Hartson, 1993):

  - Quantitative: It is a numeric data that can be measured, in order to represent the user performance in using the prototype;

- Qualitative: Qualitative data represents the opinions and improvement suggestions provided by the users during and after the prototype test;
- Objective: The objective data is the quantitative measures obtained through the observation of the users while they are interacting with the prototype;
- Subjective: The subjective data is qualitative measures, which represents the opinions of the users about the usability of the interface. Subjective measures can be obtained for example, by user questionnaire.

3.4.2 Useware Engineering

Germany organizations (i.e. GfA, GI, VDE-ITG and VDI/VDE-GMA) created the term useware in order to put stronger emphasis on users needs, working styles, requirements and preferences, as well as consider them right from the beginning in all phases of the product development process (Hussmann et al., 2011).

As previously mentioned, the development of user interfaces for interactive systems is a time consuming and costly task. To make the development process more efficient and effective, a methodical procedure with an early focus on user and task requirements is fundamental. Thus, this led to the systematic Useware Process Engineering, which focuses on understanding the user, their tasks and use of context analysis, before starting the real development. Interdisciplinary teams composed of several specialists in different areas (such as computer scientists, mechanical engineers, psychologists and designers) worked with the final users showing them prototypes from the beginning of the development phases, thereby allowing continuous and parallel evaluation as shown in the figure 5. This development process is determined by the procedure of International Organization for Standardization (ISO) 13407 (user centred design) and follows the policy of ISO 9241-110 (dialog principles).

The figure 5 illustrates the useware engineering process, it is composed by the phases analysis, structuring, design, realization and evaluation. However, the evaluation phase occurs at the same time that the others phases occur, along the entire process.

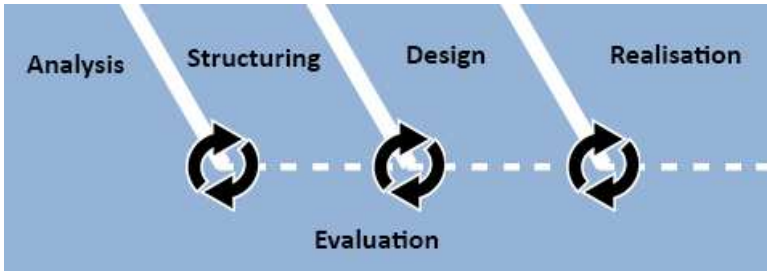

Figure 5 – Useware engeering process, adapted from (Hussmann et al., 2011).

In the software development user centered normally starts with tasks analysis. The tasks modulation is a starting point for the others development process models. The tasks can be

modulated by notations like: useML (Useware Markup Language), usiXML (User Interface Extensible Markup language) or UML (Unified Modeling Language).

In the analysis phase the context of use is defined, as well as, the characteristics and behaviours of the users through interviews, surveys, task analysis among others. Thus, the individual task models and the differences between user groups are extracted from requirements and behaviours of the users using the system. Also, during this phase issues as environmental and working conditions, as well as domain-specific context are explored. The collected data from this phase is entered, saved, analysed and exported using appropriate tools, in order to ensure that all available information is documented accurately (Hussmann et al., 2011).

The structuring phase concentrates on the junction of the user requirements and individual task models resulting of the analysis phase, in order to develop a comprehensive system, which is platform independent use model. This model has the ability to specify for example, what kinds of tasks can be performed by certain user groups at specific locations. Initially it is specified an abstract operating structure, on the basis of classification, prioritization and temporal relation of the tasks, which is saved in the XML-based Useware Markup Language (Hussmann et al., 2011).

Next is the design phase, where concepts of visualization, navigation and interaction are chosen and combined appropriately, based on the resulting data of the previous phases and the user requirements. In this phase aspects like the implementation of the use model in a specific hardware platform and the development of GUIs (Graphical User Interface) need special attention. The GUI development is composed by the following phases: graphical layout, content and behaviour. The graphical layout is related with the arrangement of the objects that compose the interface. The content refers to the information that the system provides and the behaviour is related with the dynamic responses of the system to the user actions (Bock et al., 2006). The final result of the design is to transform the design of coarse mask layouts in a fine design, which the focus is to provide an efficient support to the user, as well as, provide the information in a quick and systematic way (Hussmann et al., 2011).

The realisation phase occurs parallel to the design phase. This phase consists on the start of the implementation of the previous developed concepts into a user interface, using the hardware and software platforms that were selected in the previous phases.

As previously mentioned, the evaluation occurs parallel to these four phases, and represents a continuation of the analysis phase. This means that the development results are tested and evaluated continuously through structural or executable prototypes in all phases of the development process. The continuous evaluation allows an early detection of problems and thus reduces development costs. In the evaluation it is important to include structural aspects (like navigational concepts, among others) and not only design aspects. The users' interfaces are improved iteratively, to ensure that each user evaluation is taken into account. Modifications to the use model can be made for example, by returning from later phases to earlier ones (Hussmann et al., 2011).

In addition, it is noteworthy that the phases of the useware engineering process are structured and organized in way to respond efficiently to the user needs. This interactive process implies rapid prototyping, and each prototype is evaluated by the final users or representatives end users. The inclusion of the user since the early development phases is indispensable to avoid a system that in the end does not match the desires and needs of the user. Also, it can avoid major development costs, the more delayed the resolution of a problem encountered, greater will be the impact on all the work previously done (Hussmann et al., 2011).

3.5 Chapter Summary

In this chapter, it was addressed the definition of human-computer interaction. HCI is an important part of systems design, considering that quality of the computational system is dependent of how it is represented and usable for the users. In the interactive system development the user interface has an important role, because it is the window to view the capabilities of the system. The user interface establishes the link between the user and the system. The user performs actions on the system through the interface, and the system shows the results of these actions through the interface. The users only see the interface, this way the interface is the system for them. In the section 3.1 it was referred how important is a user interface on an interactive system. Furthermore, in the section 3.2 it was described the evolution that user interfaces have been along several decades to these days.

The concept of usability also emerged as a result of research on human-computer interaction, referred in the section 3.3. The user assumed an important role in software development. The user interface development process started to be user-centred, in order to increase the usability of the system and in turn make the final system successful. In the section 3.4 were described two of these methodologies of the user interface development process, which are: star life cycle and useware engineering. The methodologies are inherently iterative, being this way a process of self-correction. These means that is used rapid prototyping to evaluate with the final users if the system is on track. These two methodologies have several points in common, like the usability evaluation. In the star life cycle methodology, the usability evaluation assumes a central position where all other phases converge. In the useware engineering methodology the usability evaluation is performed in parallel with other phases, which means that evaluation is ongoing. Both methodologies lead to the use of rapid prototyping, in order to exist since early in the development process something to evaluate. Another similar point is the analyses phase, which is a starting point in both methodologies. However, in the star life cycle methodology does not exist an order of phases to follow. Also, in both methodologies the development of the user interface software part is different from the interaction part. In other words, there is a separation between the behaviour of the system and the implementation of that behaviour through code.

4 Machine Learning Techniques

Machine Learning (ML) field is an important subarea of artificial intelligence, which aims to imitate intelligent abilities of humans by machines (Chaudhary et al., 2012). In other words, machine learning techniques can make a computer behave independently and intelligently. As learning is the core of intelligence, an intelligent system should have a learning mechanism. Therefore, the computers should have the capability of learning by themselves, and adapt to disturbances without human intervention (Madureira et al., 2014).

Advances on areas such as: mathematics, statistics, and computer science, among others, have been allowing the rapid evolution of machine learning techniques (Alexander, 2013; Madureira et al., 2014). As result, in the last decade the use of machine learning techniques to create intelligent systems has been increasing, particularly to resolve complex problems and perform tedious human work (Alexander, 2013).

The purpose of this chapter is to provide a survey of the field of machine learning and cover the most important branches of this subject. A definition of machine learning is presented in order to understand its importance and relevance in nowadays systems. Section 4.1.1 circumscribes the types of learning commonly presented in literature (supervised, unsupervised, semi-supervised and reinforcement learning). Each of these techniques serves a different proposes, according with the type of problem, we are dealing with. For each of these learning techniques there are different machine learning algorithms that can be used. Section 4.2, describes the most commonly used (Decision tree, random forest, support vector machines, neural networks, naive bayes and k-means).

4.1 Definition of Machine Learning

Nowadays, computers are specifically defined to perform an infinite number of tasks. However, they cannot be explicitly programmed to make intelligent decisions by themselves, in order to give response to unexpected events. As learning is the core of intelligence, a system cannot be truly intelligent if it is not capable of learning. Machine learning intends to serve this purpose.

Machine learning can be defined as the field that studies computer algorithms and techniques to enable computers to learn automatically without human intervention. A machine learning system must mimic human reasoning sufficiently to support decision making. As result, they can perform actions and make intelligent decisions without being explicitly programmed to (Xue & Zhu, 2009). The learning process consists in the system acquiring knowledge by searching for patterns in data and using that data to improve the program's own understanding (Madureira et al., 2014). It is a continuous process, a learning system usually starts with some knowledge, and over time it acquires new knowledge, and as more it acquires knowledge more accurate and intelligent it becomes. Thus, it can be concluded that an intelligent system needs a period of existence to learn from experience of past tasks, in the same way that humans need. The system will continuously to self-improve and increasingly become more effective and efficient (Ayodele, 2010; Madureira et al., 2014). The figure 6 shows the basic behavior of a machine learning system. The learning part of a system acquires new knowledge from the environment, in which it operates, and updates its knowledge database with this new data (Xue & Zhu, 2009). This way, it adjusts the own actions accordingly to the new data learned, and as result, it will become progressively more accurate and intelligent.

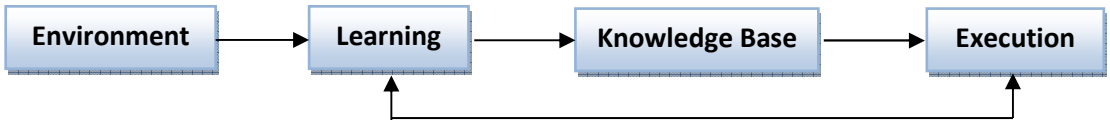

Figure 6 – Learning system, adapted from (Xue & Zhu, 2009).

Literature have been categorizing the learning process into several different categories, being the most common ones in literature: the supervised, unsupervised, semi-supervised and the reinforcement learning. Each of these techniques may use different algorithms, like: decision tree algorithm is usually used in supervised learning, on the other hand, K-Means is commonly used in unsupervised learning, reinforcement learning can use techniques like Q-Learning or Temporal-Difference learning. After, having identified the type of learning (supervised, unsupervised, semi-supervised or reinforcement learning) most suitable for a certain problem, it is necessary to choose the algorithm to use, which will be responsible for acquiring knowledge by analysing the data provided by the environment. As result, the algorithm learns how to perform important tasks by generalizing from examples. The application of machine learning techniques can be a hard task, because there are several decisions to make, such as: which classifier algorithm to use, feature selection technique, number of features to use, among others (Dittman et al., 2013). There is not an optimum decision. It may vary with the

type of problem. The execution part is the core of this entire process (Figure 6 – Learning system, adapted from (Xue & Zhu, 2009)., because it will make the system perform tasks and make intelligent choices, with base on what it learned previously. Also, it increases its own knowledge and performance when it is presented with new information (Mitchell, 1997; Mitchel, 2006).

Machine learning application has been throughout several branches of artificial intelligence such as: pattern recognition, natural language understanding, automated reasoning, intelligent robots, among others. Specific applications that have been using machine learning techniques are for example: voice and handwriting recognition, strategy games, search engines, stock market analysis, medical diagnosis, detection of credit card fraud, DNA sequencing sequences, among others (Chaudhary et al., 2012; Xue & Zhu, 2009).

4.1.1 Types of Learning

Machine Learning can be categorized into several different types of learning, being the most common ones in literature: the supervised, unsupervised, semi-supervised and the reinforcement learning, as shown in the figure 7. These categories represent how the learning algorithm works: Supervised learning algorithms are trained based on labelled data, while unsupervised learning algorithms operate with unlabelled data. The semi-supervised learning algorithms combine both techniques. On the other hand, reinforcement learning algorithms have the objective of maximizing a reward (reinforcement) (Alpaydin, 2004; Chaudhary et al., 2012; Madureira et al., 2014). Below are described in more detail this types of learning.

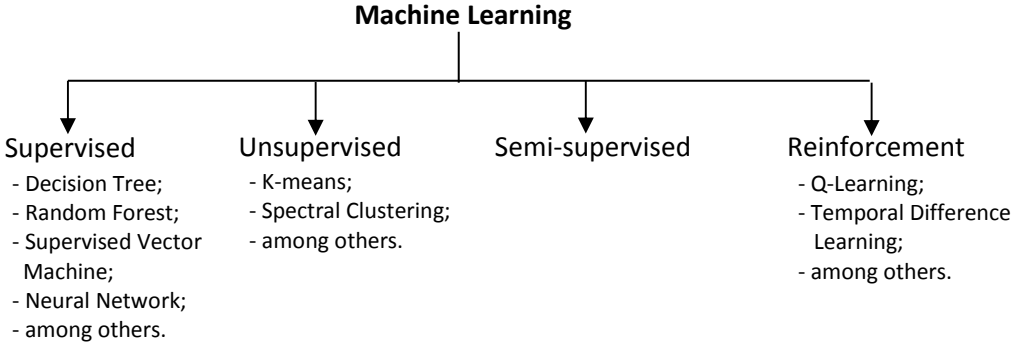

Figure 7 – Types of learning.

- **Supervised Learning**

Supervised learning algorithms learn a function from input data to result in a general hypothesis, which will predict about future new data. The input data is called training data and it consists in a set of training examples. Each example represents a possible input data associated with the desired output. The output can be a continuous value (regression), or a predicted label of the input example (classification). This algorithm will try to build the map between the input and output data, building a model which can be used to identify properties from unknown samples (Dittman et al., 2013). After the supervised algorithm is first

presented with training data, it should then be able to generalize from the presented data to new examples. In other words, it should predict the value of the function for any valid input in the future. (Ayodele, 2010; Chaudhary et al., 2012; Dittman et al., 2013). For example, an intelligent system that has the goal of predicting if a given person will be a future customer of a given service. A sample of the training data that can be used in this kind of problem is presented below.

In this training data, each example (data row) consists in a set of information about a possible customer and is labelled with yes or no, which indicates if it will be a future customer. The learning system will use this data to train and create a learning model, which should be able to make predictions in the future about new data.

Table 2 – Example of training data.

Age	Gender	Payment Method	Profession	Future customer
53	male	cheque	professor	yes
25	male	credit card	student	no
35	female	cheque	doctor	yes

There are many supervised learning algorithms, being the most common ones: decision trees, random forest, support vector machines, neural networks, among others (Chaudhary et al., 2012). In the section 4.2 of this chapter is presented a description of these algorithms.

- **Unsupervised Learning**

Unsupervised learning algorithms unlike the supervised ones build models without labelled input data. The input data is treated as a set of random features, and the unsupervised algorithm tries to discover structure in the data. In other words, it should look for patterns in the multi-dimensional feature space. This technique can also be called data clustering. Clustering is the process of portioning a group of data into a small number of clusters (groups).

As the desired output is not known, the unsupervised algorithm organizes the data in clusters according with the patterns found. Therefore, each cluster have data that are similar to each other and dissimilar to the data belonging to others clusters, and the data in each cluster share some common characteristics (Alpaydin, 2004; Ayodele, 2010; Chaudhary et al., 2012; Zhang et al., 2013). Common unsupervised learning algorithms are: K-means and spectral clustering.

- **Semi-supervised Learning**

The success of a learning process using a supervised learning algorithm heavily depends on the quality of the training data. However, it can be not available, or available in a very reduced number. Furthermore, the acquisition and selection of training data can be time-consuming, scenario-dependent and needs prior knowledge. The acquisition of labelled data has a higher

cost than the acquisition of unlabelled data, because it needs human intervention to classify each example data. On the other hand, unsupervised methods have demonstrated good results using unlabelled data, which is easier to get. As result, semi-supervised learning emerged to give response to this issue (Camps-valls et al., 2007).

Semi-supervised learning algorithms combine both techniques (supervised and unsupervised learning). It uses labelled and unlabelled data to train and test the learning algorithm. Usually, it is used a large amount of unlabelled data and a small amount of labelled data. The major advantage of this technique is that it has a reduced cost of the learning process (small amount of labelled data is needed), while having good results in the classification of new data (Camps-valls et al., 2007).

- **Reinforcement Learning**

In the reinforcement learning, the learner is a decision-making agent, which learns by interacting with an environment. The agent produces certain actions, which affect the state of the environment, trying to solve a problem. These actions result in the machine receiving a reward (or penalty). The goal of a reinforcement learning algorithm is learning how to act in way that maximizes the future rewards it receives. After several trial-and-error runs, the agent should have learned the best policy to use in the future (Alpaydin, 2004; Chaudhary et al., 2012; Panait & Luke, 2005). For example, an intelligent system that plays chess cannot use a supervised learner. First, it is impossible have a training data with many games and that indicate the best move in each situation. Also, there is no such thing as best move, because it depends on the following moves. A sequence of moves is good if after playing them we win the game (Alpaydin, 2004). Reinforcement learning fits this kind of problem. It will learn over time how to play chess by producing a sequence of actions and observe their consequences.

Two common reinforcement learning methods referred in literature are Q-Learning and Temporal-Difference learning. The Q-Learning learns the utility of performing actions in states, while the Temporal-Difference learning learns the utility of being in the states themselves (Panait & Luke, 2005).

4.2 Machine Learning Algorithms

There are different ways that an algorithm can model a learning problem. The choice of which algorithm to be used should depend on the domain of their application (Chaudhary et al., 2012). Thus, after identified the type of machine learning problem (supervised, unsupervised, semi-supervised or reinforcement learning), should be chosen the algorithm to implement the learning model. All the algorithms can be valuable, so this choice should depend on the characteristics of situation: the kind of problem being solved, how much time is dispensed to solving it, what type of data the model will be working with (Fathima & Hundewale, 2012). In this section it is described some of the most popular machine learning algorithms, namely: Decision tree, random forest, support vector machines, neural networks, naive bayes and k-means.

4.2.1 Decision Tree

Decision tree is a supervised machine learning algorithm and one of the most widely used. This type of algorithm has very powerful methods on inferring classification rules (Madureira et al., 2014). Decision tree learning uses a set training data to build a classifier to predict the class label of a new case (Elomaa & Malinen, 2003). It is represented by an inverted tree composed by several nodes, as it can be seen in the figure 8. Each node will be expanded until the stopping criterion is satisfied. Each node that is expanded results in a new sub-tree.

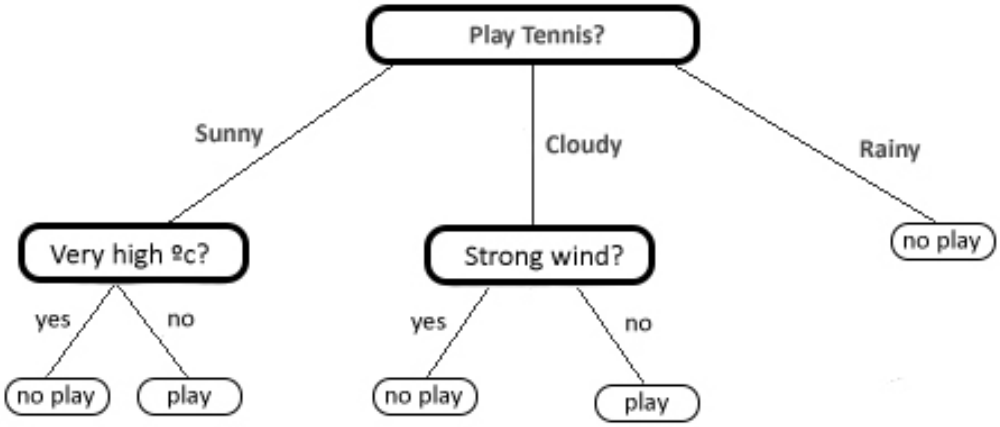

Figure 8 – Decision tree example.

Each interior node of the tree implements a decision rule, which splits the data in two or more partitions. The size of a node is the number of training data elements in its subset. Each branch node represents a possible decision, and each leaf node (terminal node) represents the final classification or decision (Madureira et al., 2014; Mallios et al., 2011; Vijaya et al., 2009).

The two major advantages of decision trees over other classification methods are: First, the decision rules can deal with both numerical and non-numerical data, which allows the application of this type of algorithm to a wide range of problems. The other advantage is that decision trees are simple to understand and interpret. A user can easily follow the sequence of decisions from the root node to the leaf nodes, and interpret the intrinsic rational of the classification algorithm (Madureira et al., 2014).

4.2.2 Random Forest

Leo Breiman proposed the random forest algorithm, for classification and regression. This algorithm generates a set of decision trees predictors, resulting in a *forest* model (Boulesteix et al., 2012). Each tree has the same representation and logic that was described in the previous section, with one exception: for each split only a random subset of attributes is available (Breiman, 2001). Random forest emerged because one of the disadvantages of decision trees is that they do not generalize well, due to the high variance of its predictions.

Random forest overcomes this issue by taking the average of the predictions of several trees (Madureira et al., 2014).

4.2.3 Support Vector Machines

Support Vector Machine (SVM) is a supervised algorithm based in statistical learning theory, and has been successfully applied to a wide range of pattern recognition problems (Vijaya et al., 2009). The SVM represents the examples as points in space. As can be seen in the figure 9, this algorithm builds a binary classifier, by constructing a hyperplane that separates examples from different classes. The different classes (figureFigure 9 – Support vector machine example, adapted from (Ben-Hur et al., 2008).) are represented by the colors red and blue respectively. The distance between these hyperplanes should be maximized (maximum margin), in order to preserve the greatest distance between each classes (Dittman et al., 2013; Madureira et al., 2014; Vijaya et al., 2009).

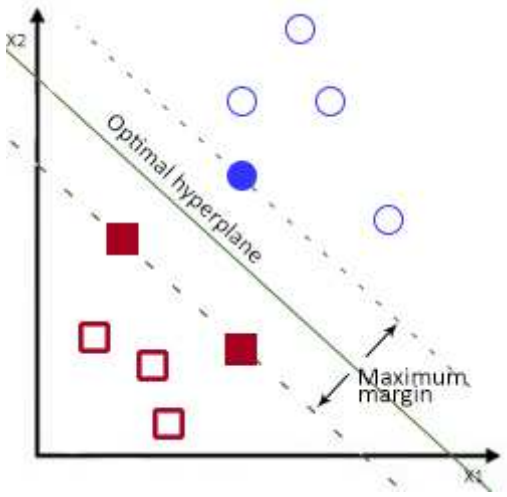

Figure 9 – Support vector machine example, adapted from (Ben-Hur et al., 2008).

4.2.4 Neural Networks

The neural networks algorithm is another supervised algorithm, which is based on biological neural networks of the brain. The Multilayer Perceptron (MLP) network, which can be seen in the figure 10, is the most widely used neural network classifier (Vijaya et al., 2009). It uses nonlinear tools to map a set of input data to a set of outputs, in order to find patterns in data (Madureira et al., 2014; Mallios et al., 2011).

As it can be seen in the figure 10, the MLP can be represented by a stack arrangement of layers, which are composed by a group of artificial neurons that have the goal of processing the information. The layers in the middle (between the input and output layer) are called hidden layers. The number of hidden layers of a MLP can be defined by the user, as well as, the number of neurons that each middle layer should have.

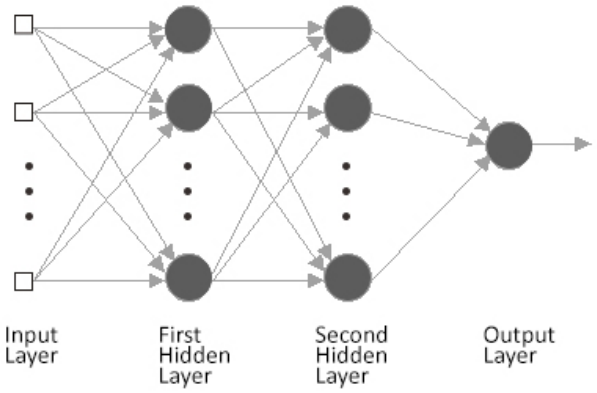

Figure 10 – Neural networks example, adapted from (Madureira et al., 2014).

The number of neurons in the input layer correspond to the number of attributes (input data), and the number of neurons in the output layer corresponds to the number of labels/classes in the classification problem. The complexity of the MLP network is related with the number of layers and the number of neurons in each hidden layer (Madureira et al., 2014; Vijaya et al., 2009).

The learning process in a MLP algorithm is done using back-propagation, which can be defined in two steps (Madureira et al., 2014):

1. The data is propagated into the network and generates an output that will be compared with the correct answer;
2. The information about evaluation of the output is then propagated backwards into the network to adjust its weights, in order to reduce the value of the error

These two steps are repeated a number of times, which is defined by the user, and then the network converges to a state with the smallest error. As result, the MLP is capable of producing an output from new data, which would correspond to the expected output from the approximated mapping (Madureira et al., 2014).

4.2.5 Naive Bayes

Naive Bayes is one of the most simple probabilistic classification algorithms, and can be a very effective classifier when applied in supervised learning problems. It has been used successfully in several applications of information processing, such as: natural language processing, information retrieval, among others (Vijaya et al., 2009). This probabilistic classifier is based on Bayesian theorem, which considers a strong independence assumption (naive assumption). In other words, it assumes that the presence of a given feature of a class is independent to the presence of any other feature (Fathima & Hundewale, 2012; Madureira et al., 2014). Classification in Naive Bayes model is performed by applying the Bayes rule to each attribute of the model and the probability over an independent class variable is calculated (Mallios et al., 2011).

4.2.6 K-Means

The K-Means algorithm is commonly used in unsupervised learning. It joins objects together that are similar to each other and dissimilar to objects that belong to other groups, which can be called clustering. Therefore each object is assigned to one cluster (Zhang et al., 2013). The figure 11 describes the operation mode of the K-Means algorithm.

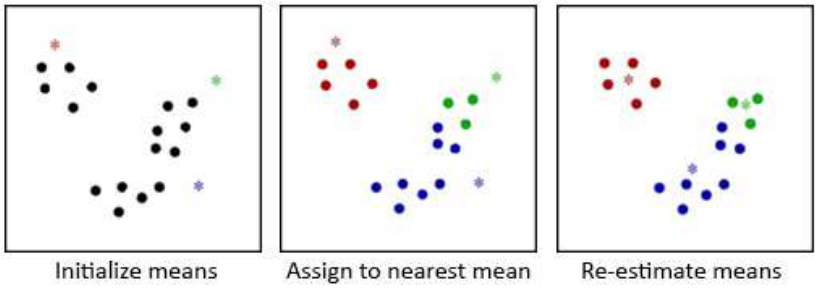

Figure 11 – K-Means example, adapted from (Kumar, 2002).

In the figure the objects are represented by black dots. The operation mode of this algorithm can be described in three major steps:

1. The K-Means determines a set of K points, called centres or centroids, each K point represents the center of each cluster;
2. The algorithm assigns each object to nearest mean (forming a cluster);
3. It moves the mean to center of its cluster.

The last two steps will be iterated several times, and then the algorithm will converge to a local minimum of the objective function in a finite number of iterations, which can be defined by the user. The figure below shows the clusters after several iterations.

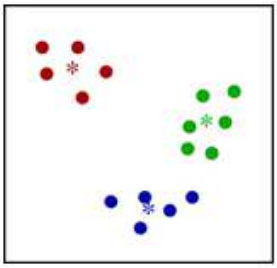

Figure 12 – K-Means example after several iterations, adapted from (Kumar, 2002).

In general, it can be concluded that the quality of the clustering and the number of iterations for convergence depend on the initial choice of centroids (MLF, k-means, k-means 2, kmeans3). Some arrangements of the K points can cause problems, such as: clusters with different sizes, densities, non-globular shapes; outliers (objects that are not relatively close to any cluster); empty clusters (Zhang et al., 2013).

4.3 Chapter Summary

This chapter provided a review on the machine learning field, as well as, machine learning techniques. Machine learning is a valuable tool, it can make a system extract and learn useful pieces of information from certain data, in order to support decision making (Mallios et al., 2011). This way, computer systems could have the ability to perform tasks and make decisions without being explicit programmed to. Therefore, an intelligent system could reduce human work and as result, it can increase organizations agility, productivity and effectiveness (A Madureira, Santos, et al., 2014). Presented this point of view can be concluded that machine learning is a valuable tool. By improving the machine learning field, we can make machines' intelligence be close to or surpasses the humanity's intelligence level (Xue & Zhu, 2009).

The learning is a continuous process so that the intelligent level of the system will be continuously increased. Section 4.1.1 presented four types of learning commonly referred in literature (supervised, unsupervised, semi-supervised and reinforcement learning). Furthermore, section 4.2 presented several different machine learning algorithms (Decision tree, random forest, support vector machines, neural networks, naive bayes and k-means), which can bring intelligence to a system. It cannot be stated that one type of learning or one algorithm performs better than another. Each has strengths and weaknesses and serves different purposes. Therefore, the performance of any machine learning technique and algorithm depends upon the application domain and the application requirement. Scheduling system areas have great potential for improving performance by using machine learning techniques.

5 Dynamic Scheduling Problem

Scheduling plays an important role in manufacturing systems. It shares resources to produce several different products in the same time period, with the goal of optimizing the organization objectives. However, current manufacturing scheduling systems have a dynamic nature, they are subject to a variety of unexpected events, which may disturb the working conditions. Therefore, there is a need to incorporate these dynamic events into the scheduling process, in order to ensure feasibility of the scheduling plan that the manufacturing system is following (Cowling et al., 2004; Pinedo, 2005; Priore et al., 2001; Vieira et al., 2003).

Static scheduling is not effective any more in this type of problem. This chapter provides a review of the state of the art of research in dynamic scheduling. First, it is described a generic manufacturing environment and the role of its scheduling function, outlining the limitations of the static scheduling approach in the presence of real-time disturbances. Also, it presents the terminology and notation that will be used throughout this work. In the section 5.2.1 it is presented the dynamic scheduling approaches most addressed in literature: reactive and predictive-reactive scheduling. This work focuses mainly in the predictive-reactive scheduling approach. As result, this chapter addresses the problem of when to reschedule, and what rescheduling strategies to use to incorporate real-time events.

5.1 Definition and terminology of scheduling

Scheduling is a decision making process used on a regular basis in many manufacturing organizations. It uses mathematical techniques and heuristic methods to allocate limited resources to jobs under a number of constraints, in an efficient manner. The allocation of resources must be done in way that the organization optimizes its objectives and achieves its goals. Generally, these objectives can be: minimize the time to complete all jobs, minimize the number of jobs that are completed after its due dates, or maximize the occupation of resources, among many others. Overall, a detailed schedule plan helps the organizations

maintain efficiency and control of the operations. The resources can be for example the machines in a workshop, runways at an airport, or processing units in a computer environment, among others. The jobs can be the operations in a workshop, take-offs and landings at an airport, or computer tasks that have to be executed, amongst others (Baker et al., 2009; Ouelhadj et al., 2007; Pinedo, 2005; Priore et al., 2001; Vieira et al., 2003).

However, in most real world manufacturing problems, resources can be described as machines, and the orders need to be executed in these machines. Orders can be referred as jobs or tasks. Each job is characterized by a priority level, an earliest possible starting time and a due date. Also, each job can be composed by a single operation or a set of operations, which can be seen as sub-jobs. Each operation uses one of the machines for a period of time. Each machine can process just one operation at a time, and the processing of an operation cannot be interrupted. The operations must be processed in a predefined order or sequence. The jobs also can be subject to precedence constraints, for example, some jobs cannot start until other jobs have been completed (Madureira et al., 2010).

In practice, it is necessary take into account that current manufacturing environments have a dynamic nature, because unexpected events and perturbations on working conditions can occur over time, which can have a major impact in the scheduling plan and sometimes, it is even necessary to reschedule the entire plan (Baker et al., 2009; Cowling et al., 2004; Jensen, 2001; Pinedo, 2005; Priore et al., 2001; Yingzi et al., 2009). The section 5.2 gives a more detailed review about dynamic scheduling problems.

5.1.1 Terminology and notation

The following terminology and notation will be used across this document. It is divided into the categories static and dynamic. The static data is already defined before the scheduling, this means that static data is not dependent of the schedule. Otherwise, dynamic data is defined according to the scheduling plan. A $N \times M$ scheduling problem consists of N jobs and M machines. In the following table the subscript j refers to the job j, and the subscript i refers to the machine i (Pinedo, 2005). The following data pertain to job j.

Table 3 – Terminology and notation of scheduling systems (Pinedo, 2005).

Data	Designation	Notation	Description
Static	Processing time	p_{ij}	Represents the time that the job j spends on the machine i.
	Release date	r_j	It is the time that the job j arrives at the system. In other words, the earliest time at which job j can be processed.
	due date	d_j	The due date of the job j represents the date when the task must be completely executed. If the job is not completed until its due date, a penalization is incurred. Sometimes the job the due date must be absolutely met, then it is a deadline.
	Weight	w_j	The weight of the job j represents the priority of the job relative to other jobs in the system. It can also represent the cost of keeping the job j in the system for one time unit.
Dynamic	Starting time	S_{ij}	The starting time is the time when a job j starts its processing on machine i.
	Completion time	C_{ij}	Completion time is the time when job j is completed on machine i.

5.2 Dynamic Scheduling

As described in the section 5.1, scheduling aims to plan the use of resources in the processing of tasks, trying always to improve the utilization ratio of the resources. Static scheduling assumes that all tasks will be processed on time, all equipment is in normal working state, and that no disturbances will occur. However, this is not always true in the majority of real world manufacturing environments. In the daily activity of organizations, they constantly have unexpected events and disturbances on working conditions and requirements. These dynamic events and disturbances can be for example: a busy or broke machine, energy sources shortage, urgent job arrival, job cancelation, due date change, delay in the arrival, shortage of materials, and change in job priority, among others.

Dynamic events must be taken into account, since they may have a major impact on the schedule, they can change the system status and affect performance. Therefore, static scheduling is not the most adequate approach, since dynamic events will disorder the scheduling plan and make it ineffective. Manufacturing systems require immediate response to these dynamic events, using a real-time scheduling method. As result, rescheduling is mandatory to minimize the effect of such unexpected events in the performance of the system. Rescheduling can be defined as the process of updating an existing scheduling plan in response to the dynamic events.

Dynamic scheduling should have the ability to efficiently and effectively adapt, on a continuous basis, the current schedule to the referred events. In order to rapidly adapt the manufacturing system to market and environmental changes in an efficient and cost-effective way. Therefore, an effective manufacturing environment needs an effective schedule, which can result in improved on-time delivery of products/services, improved quality, reduced costs and increased productivity.

The main difference between dynamic and static scheduling lies on the used algorithm for scheduling, the dynamic one requires robustness and efficient reactivity to the changing environment. Also, a dynamic scheduler leads to other important decisions, namely if and when rescheduling should happen. To answer this question, literature have been discussing strategies for rescheduling manufacturing systems, as well as, techniques for solving constraint violations that can occur when unexpected events happen (Baker et al., 2009; He et al., 2008; Jensen, 2001; Pinedo, 2005; Priore et al., 2001; Sabuncuoglu et al., 2009; Vieira et al., 2003; Yingzi et al., 2009).

5.2.1 Dynamic Scheduling Approaches

When dynamic events occur and cause significant performance deterioration of the scheduling plan, rescheduling should be done to reduce the impact of these unexpected events. Rescheduling consists in generating a new schedule to incorporate the changes in the environment and part of the preschedule (the schedule plan prior to occurrence of a dynamic schedule). Ideally this schedule should have an implementation cost as low as possible (Cowling et al., 2004; Jensen, 2001; Vieira et al., 2003).

(Vieira et al., 2003) describes two main strategies (reactive scheduling and predictive-reactive scheduling) to control a manufacturing system in a dynamic rescheduling environment that have dynamic jobs arrivals. Rescheduling strategies have the goal of answering the question of when scheduling plans should be generated. Both of these rescheduling strategies can be used in any rescheduling environment. However, dynamic rescheduling environments are the most common case in manufacturing systems. Also, in practice the predictive-reactive rescheduling strategy is the most used approach, and there are a variety of rescheduling policies. Rescheduling policies specify the events that trigger the rescheduling, in other words, specifies the method used to revise the existing schedule. The policy can specify different methods for different situations. Also, the policies can specify several parameters like for example: the length of the rescheduling period. Below is described the two mostly addressed rescheduling strategies in literature (Sabuncuoglu et al., 2009; Vieira et al., 2003):

- **Reactive Scheduling**

In reactive scheduling the control of a manufacturing system is performed in real-time, because decisions are made based on the current state of the manufacturing system. This strategy does not create scheduling plans. Instead, jobs are dispatched when necessary, based on the current information available in the system. It uses dispatching rules or other heuristics

to control a manufacturing environment without a scheduling plan. For example, when a resource becomes available, it uses a dispatching rule to choose the next job to be performed from among the jobs in its queue. The dispatching rule can sort the jobs by some criteria to choose the job;

- **Predictive-Reactive Scheduling**

Predictive-reactive scheduling is an iterative process, commonly used to rescheduling dynamic manufacturing systems. Unlike the previous strategy, a scheduling plan is produced. However, this scheduling plan may be rescheduled to give response to real-time events, in order to minimize its impact on the system performance. Rescheduling can occur frequently, or it can just be a single revision of the schedule. To implement a predictive-reactive scheduling strategy a rescheduling policy is needed. A policy determines when and how to react to disturbances. Following is presented three most common types of rescheduling policies that have been proposed in literature: periodic, event-driven, and hybrid (Sabuncuoglu et al., 2009; Vieira et al., 2003):

  - **Periodic**

A periodic policy reschedules the scheduling plan on a continuous basis. This periodic approach is regularly used in manufacturing environments where there is no data available to monitor the state of the plan in real time. Therefore, a scheduler will generate scheduling plans at regular intervals, based on all available information that the scheduler could gather. Therefore, a scheduling plan will then be implemented and not revised until the next period begins. As result, this approach ensures more stability than constant rescheduling. However, following an established plan which may not reflect significant changes that may occur in the system status can compromise performance. Determine the optimal rescheduling period is another difficult decision when using this type of policy;

  - **Event-Driven**

In an event-driven rescheduling policy, rescheduling occurs when a dynamic event happens. This approach is generally used in static systems to reschedule the system when a resource fails, for example. Rescheduling every time that a dynamic event occurs can lead to too much time spent. Also, it requires a fast and reliable electronic data collection to quickly capture new events. As result, in systems where dynamic events are constantly happening, the system can stay in permanent state of rescheduling. This situation causes in the manufacturing system low stability and higher computational costs;

  - **Hybrid**

A hybrid rescheduling policy is a combination of the two previous policies. It can reschedule the system periodically and also when a major dynamic event happens, such as: a machine breakdown, arrival of an urgent job, among others.

The study of (Vieira et al., 2003) shown that the rescheduling frequency can significantly affect the system performance. Rescheduling when an urgent job (one with a tight due date) arrives is useful. However, very frequent rescheduling (short rescheduling periods) do not improve the system performance significantly. A lower rescheduling frequency decreases the number of setups (i.e. reconfiguration of resources), reducing unproductive time wasted on setups. Otherwise, higher rescheduling frequency can make the system react more quickly to disturbances but the number of setups may be increased.

5.2.2 Rescheduling Methods

As rescheduling is a necessary reaction to disturbances. In this section are described the methods used in predictive-reactive scheduling to generate or update scheduling plans. These methods have the goal of updating the schedules in response to disturbances, generating a robust scheduling plan. Generating robust schedules is an attempt to maintain good system performance with simple schedule adjustments (Jensen, 2001; Vieira et al., 2003):

When a scheduling plan is generated, manufacturing operations begin. Supervisors of a manufacturing system want to follow the scheduling plan. However, in practice this is almost impossible. Operators will probably deviate from the scheduling plan. Small deviations from schedule start times and end times are expected and usually ignored. When unexpected events occur and disrupt the initial schedule, larger deviations happens. There is where scheduling repair is needed. Even if the supervisors do not explicitly update the schedule, schedule repair occurs automatically. Literature describes three common methods to repair a scheduling plan that is no longer feasible due to the disturbances occurred: Right-shift rescheduling, regeneration and partial rescheduling (Jensen, 2001; Vieira et al., 2003).

- **Right-shift Rescheduling**

Right-shift rescheduling is the simplest and fastest way to solve a rescheduling problem. It keeps the processing order of the preschedule, but postpones the remaining operations by the amount of time required to make the schedule feasible again;

- **Regeneration**

Regeneration reschedules the entire set of jobs not processed before the rescheduling point, including those not affected by the disturbance. The main disadvantage of this method is the excessive computational cost and unsatisfactory response time. To overcome this problem the prescheduled plan can be used to solve the scheduling problem every time a new job arrives;

- **Partial Rescheduling**

In partial rescheduling when a disturbance in the scheduling system occurs, only the operations affected directly or indirectly by the disturbance are rescheduled. This method preserves the initial state of the scheduling plan as much as possible. Thus, schedule stability is maintained.

5.2.3 Performance Metrics for Dynamic Rescheduling

Performance measures such as makespan, flowtime, earliness and tardiness are the most used measures in practice. However, performance measures related to dynamic manufacturing environments such as utility, stability and robustness have been also described in literature. These measures allow the study and analyse the impact of dynamic events in a scheduling plan.

In a dynamic scheduling process an initial scheduling plan is generated, when an unexpected event or disturbance happens, this schedule is partly or completely updated, forming a new scheduling plan. This schedule can be substantially or drastically different from the previous schedule. The measures robustness and stability are related with this difference.

Robustness is a desirable feature in any dynamic scheduler, it is related to the insensitivity of scheduling performance to disturbances. The performance of a robust plan remains valid in the face of disturbances. Stability is related with the difference between the initial plan and rescheduling plan. A stable rescheduling plan does not deviate much from the initial schedule. (Cowling et al., 2004; Jensen, 2001; Sabuncuoglu et al., 2009; Vieira et al., 2003).

Cowling et al. (2004) defined these three performance measures: *robustness*, *stability* and *utility*.

Utility is defined by "*the difference between the value of the objective value $F_{dynamic}$ of the new schedule $S_{dynamic}$ after taking into account the real-time information E which arrives at time t, and the objective value F_{static} of the initial schedule S_{static} before taking into account real-time information*" (Cowling et al., 2004). Utility is expressed by:

$$Utility(S_{static}, S_{dynamic}, E, t) = F_{dynamic} - F_{static} \quad (1)$$

Stability is obtained from "*the sum of the absolute difference between the original completion time C of each coil of the original schedule and the new completion time C' after the occurrence of the real-time event for each coil*" (Cowling et al., 2004). Stability is expressed by:

$$Stability(S_{static}, S_{dynamic}, E, t) = \sum_{i=1}^{N} |C_i - C'_i| \quad (2)$$

Robustness of a schedule plan S "*combines the maximization of the efficiency $utility$ and the minimization of the deviation $stability$ from the original schedule*" (Cowling et al., 2004). The robustness is expressed by:

$$Robustness(S) = r \times Utility - (1 - r) \times Stability \quad (3)$$

r is a real number in the interval [0, 1].

Several contributions has been proposed in literature to dynamic scheduling resolution (Madureira et al., 2010).

5.3 Chapter Summary

In this chapter it was provided a review of the literature on dynamic scheduling. From this review it can be concluded, that current real world manufacturing environments are sensitive to disturbances (a busy or broke machine, energy sources shortage, urgent job arrival, job cancelation, due date change, delay in the arrival, shortage of materials, change in job priority, among others). Static scheduling is not effective in this type of problem, because these disturbances can disorder the scheduling plan making it not feasible any more. Therefore, rescheduling is an important and necessary part of managing a dynamic manufacturing system, with the goal of minimizing the effect of these disturbances. Rescheduling allows an immediate response to these dynamic events, by using a real time scheduling method (Baker et al., 2009; He et al., 2008; Jensen, 2001; Pinedo, 2005; Priore et al., 2001; Sabuncuoglu et al., 2009; Vieira et al., 2003; Yingzi et al., 2009).

Researchers have done a great effort on developing methods to generate optimal scheduling plans. In the section 5.2.1 were presented the most common dynamic scheduling approaches found in literature: reactive and predictive-reactive scheduling. These approaches help a manufacturing system be more productive and efficient. Predictive-reactive scheduling includes three types of policies: periodic, event-driven, and hybrid rescheduling. A policy determines when and how to react to disturbances. In a scheduling system that uses a periodic policy, a schedule plan is revised or updated periodically over time. A scheduling system that uses an event-driven policy, rescheduling will occur when certain dynamic events happen. On the other hand, a scheduling system that uses a hybrid policy, rescheduling happens periodically unless a certain dynamic event occurs.

Section 5.2.2 presented the three most common rescheduling methods: right-shift scheduling, regeneration, and partial rescheduling. Right-shift postpones the remaining operations by the amount of time it needs to make the schedule feasible again. Regeneration reschedules the entire set of jobs. Finally, partial rescheduling reschedules only the operations that are affected by the dynamic event. Lastly, this chapter addressed the performance measures: utility, stability and robustness. These measures have the goal of analyse the impact of dynamic events in a scheduling plan.

6 Evolutionary Computation and Hyper-Heuristics

Evolutionary Computation is the study of computational systems which use ideas from natural evolution to solve complex computational problems (M. Mitchell & Taylor, 1999). In nature, evolution is mostly determined by natural selection of different individuals competing for resources in the environment (Abraham, 2005). Several computational problem solvers bio-inspired have been proposed in literature, namely Particle Swarm Optimization, Ant Colony Optimization and Evolutionary Algorithms. Section 6.1 of this chapter will provide a survey on the field of evolutionary computation, addressing different evolutionary algorithms, such as: genetic algorithm, evolution strategies, evolutionary programming, and genetic programing. This algorithms have been successfully applied to solve complex problems, in areas such as: optimization, learning or design. The genetic algorithm will be described in more detail in section 6.2 , since it is the main focus of this master thesis and one of the most referred evolutionary algorithms in literature. In addition, section 6.3 will address the definition and concepts of hyper-heuristics, as well as, a classification of different types of hyper-heuristics.

6.1 Definition and Concepts of Evolutionary Computation

Evolutionary Computation (EC) is an area of computer science that is inspired from natural evolution to solve complex computational problems. Artificial intelligence researchers believed that the rules that would confer intelligence to a system could be encoded top-down way. However, this is a very complex task. The best route to solve complex computational problems is through a bottom-up paradigm, which consists in providing only very simple rules and a means for the system to adapt. This way, an approach to solve this type of problems consists in using an algorithm that searches for the most suitable solution to a certain

problem in a large space of potential solutions (M. Mitchell & Taylor, 1999). Evolutionary computation techniques have been successfully applied to numerous complex problems from different domains, such as: optimisation, learning, automatic programming, bioinformatics, and so on (Abraham, 2005; M. Mitchell & Taylor, 1999).

Evolutionary computation applies the principle of evolution observed in the nature. In nature, evolution is determined by natural selection, where the best individuals tend to survive and reproduce. This way, will be generated new individuals with better potential than their parents, if they inherit the best characteristics (genes) of each parent. There are several approaches to evolutionary computation, which will be described later. But, a better and more general term for such approaches is Evolutionary Algorithms (EAs) (Abraham, 2005; M. Mitchell & Taylor, 1999).

An evolutionary algorithm follows step by step the process that mimics the principles of natural selection and survival of the fittest individual (Michalewicz, 1996). FigureFigure 13 – Working flow of an evolutionary algorithm, adapted from (Abraham, 2005). describes the working flow of an evolutionary algorithm, which is conceptually simple. The first step is to generate a population of candidate solutions. From this population some individuals are selected to reproduce, in order to form new solutions. The best individuals have a higher probability of being selected. Reproduction is done by applying a reproduction operator (crossover), which recombines two individuals in a population to generate one or more new individuals (offspring). Also, it can be applied a genetic mutation to some individuals, in order to create diversity in the population. To each candidate solution is evaluated the fitness value, which represents how close a solution is from the optimal solution. Next, is applied a selection strategy to determinate which solutions will be maintained into the next generation. This process is repeated for several generations until some stopping criteria have been reached. As the evolutionary algorithm searches through a space of possible solutions to find the near optimal one, it can be successfully applied in optimization processes (Michalewicz, 1996). The table 4 presents a terminology summary used in evolutionary algorithms.

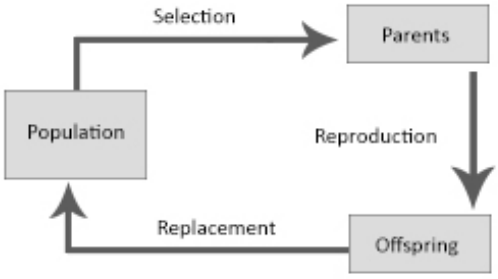

Figure 13 – Working flow of an evolutionary algorithm, adapted from (Abraham, 2005).

Table 4 – Terminology of a Genetic Algorithm.

Designation	Description
Individual	Candidate solution.
Chromosome	Representation of the candidate solution (string).
Population	Set of chromosomes.
Generations	Iterations.
Gene	Characteristic of a chromosome.

Therefore, it can be concluded that an evolutionary algorithm is a population-based and generate-and-test algorithm. To successfully solve a certain real-world complex problem using an evolutionary algorithm, it should have the following characteristics (Michalewicz, 1996):

- A genetic representation of possible solutions to the problem;
- A way to generate an initial population;
- A way of evaluate and rate each candidate solutions, in terms of their fitness;
- Genetic operators to create new solutions;
- Values for the parameters: population size, number of generations, among others.

There are several different evolutionary algorithms, as can be seen the figure 14 (Abraham, 2005) : Genetic Algorithms (GA), Evolution Strategies (ES), Evolutionary Programming (EP) and Genetic Programming (GP).

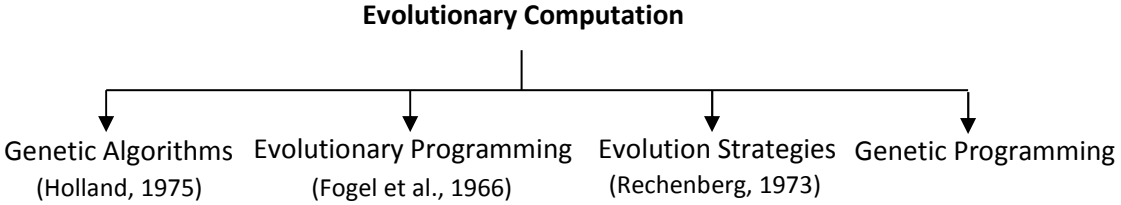

Figure 14 – Types of evolutionary algorithms.

All of these algorithms are very similar and simulate biological evolution like natural selection, in which individuals evolve by a process of selection, recombination and mutation reproduction, in order to produce better individuals (Chaudhary et al., 2012). However, they have some variants in relation to historical backgrounds, representations, genetic operators, and selection methods, which are described below. The genetic algorithms will be described in more detail in section 6.2 of this chapter, since they are the main focus of this master thesis and one of the most used evolutionary algorithms.

Genetic Algorithms were proposed by (Holland, 1975) for adaptive search and optimisation, simulating Darwinian evolution. It is the most popular type of evolutionary algorithm, and addresses three major steps: Selection, reproduction and replacement. In the selection step the fitter individuals have a greater probability of being selected to reproduce. Reproduction

is performed by applying genetic operators, in order to create a new generation. Finally, the current generation will be replaced by new individuals. This algorithm emphasises the role of recombination (crossover). Mutation is only used as a background operator.

Evolutionary Programming were first proposed by (Fogel, 1966) for simulating intelligence. It is similar to GAs, but the offspring are generated in a different way. In the reproduction process, this algorithm does not use recombination, only the mutation operator.

Evolution Strategies were first proposed by (Rechenberg, 1973) for numerical optimisation. The first version of this algorithm considers only two individuals: a parent and an offspring, which are represented by real-valued vectors. Also, they use recombination and self-adaptive mutations.

Genetic Programming were first used by (Garis, 1990) to indicate the evolution of artificial neural networks. However, it was used later to indicate the application of GAs to the evolution of computer programs. In GP the candidate solutions (individuals) are in form of computer programs and they are represented by trees, especially Lisp expression trees. The fitness of each individual is evaluated by their ability in solving a computational problem. Reproduction is performed by crossover and mutation operators.

6.2 Genetic Algorithms

Genetic Algorithms are a class of evolutionary algorithms involving search and optimization. A Genetic Algorithm combines genetic and the theory of Darwin on natural selection (Holland, 1975), in order to have the ability to find the solution for a complex problem without the need of derivating that information.

“Darwin's general theory presumes the development of life from non-life and stresses a purely naturalistic (undirected) "descent with modification". That is, complex creatures evolve from more simplistic ancestors naturally over time. In a nutshell, as random genetic mutations occur within an organism's genetic code, the beneficial mutations are preserved because they aid survival -- a process known as "natural selection." These beneficial mutations are passed on to the next generation. Over time, beneficial mutations accumulate and the result is an entirely different organism (not just a variation of the original, but an entirely different creature).” (“Darwin’s Theory of- A Theory in Crisis Evolution,”)

In 1975, John Holland presented for the first time the Genetic Algorithm, which has been successfully applied in optimization problems. This algorithm follows the major ideas of the Darwin’s theory (“Darwin’s Theory of- A Theory in Crisis Evolution,”). It performs simulated evolution on populations of individuals. As in nature, a GA solves the problem of finding the better individuals by manipulating their genetic material. Therefore, in a GA there is a permanent competition between individuals, and the fitter individuals will survive and pass their genes to the offspring, in order to increase the quality of future generations. In sum, it is

an iterative algorithm that deals with genes, chromosomes and populations, in order to find a near optimal solution to a certain problem.

This algorithm starts with an initial set of random (or not) candidate solutions, called population. It will iteratively refine the initial population until certain stopping criterion has been met. Each individual in the population is a candidate solution to the problem at hand and is called a chromosome, which is encoded on a particular representation scheme. These chromosomes will evolve through successive iterations, called generations. The search for the near optimal solution is guided by the fitness value of each individual. Therefore, in each generation the fitness of each chromosome is evaluated and assigned to each individual, this value indicates how close a candidate solution is to an ideal solution. Afterwards, the chromosomes with the best fitness value are selected from the pool of individuals to reproduce. The fitter individuals have a major chance of being selected. Once the selection process is complete, new individuals are formed by reproduction. The reproduction is made by applying genetic operators, such as: crossover and mutation, in order to produce one or more new chromosomes, which will form a new generation. In the process of crossover is combined the genetic characteristics of two individuals, which will create one or more new individuals. In the mutation process, one or more characteristics (genes) from an individual are randomly selected and changed, in order to provide greater diversity in the population. This process of evaluation, selection, and application of genetic operators is repeated for several generations, until it reached the stop criteria. The stop criteria can be for example:

- Maximum number of generations;
- Limit of the processing time;
- Evolving the algorithm until no significant improvement is found in the solutions over a given number of consecutive generations;
- Find a solution close enough to the ideal solution;
- Until there is no change in the standard deviations of the fitness values between two consecutive generations.

The two main components of a Genetic Algorithm, are the encoding schema and the evaluation function, also called fitness function. Each solution should be encoded, based on a particular representation scheme, called chromosome. The fitness function evaluates the quality of each chromosome. In the following section is described the entire work flow of a Genetic Algorithm.

6.2.1 Working Flow of a Genetic Algorithm

The Genetic Algorithm is a robust intelligent meta-heuristic suitable for optimization problems. It has the goal of searching for the near optimal solution in a large search space of possible solutions to solve the problem at hand (Gkoutioudi & Karatza, 2012). Code 1 is presents the algorithm of a Genetic Algorithm.

```
begin
t ← 0
initialize P(t)
evaluate P(t)
        while (not termination condition) do
        begin
        t ← t + 1
        evaluate P(t)
        select P(t) from P(t − 1)
        crossover P(t)
        mutation P(t)
        end
end
```

Code 1 – Genetic algorithm.

The Genetic Algorithm maintains a population of several individuals, $P(t) = \{x_1^t, \dots, x_n^t\}$ for iteration t, generation t. The algorithm starts with an already initialized population $P(0)$, this initialization can be random or not. Each individual in the population is a candidate solution to solve the problem at hand, and is represented as some scheme S. To each individual in the population x_i^t is evaluated the fitness value. This value represents the quality of the individual. The fittest individuals are selected to reproduce (crossover step (c_j)), this way a new population (iteration $t + 1$) is formed. Reproduction is performed by combining parts of the selected individuals ($c_j: S \times \dots \times S \rightarrow S$). Some individuals of the new population undergo the mutation step ($m_i: S \rightarrow S$), which alters one or more individual' genes, in order to create some variety in the population. When the stopping criteria is achieved the algorithm stops, and the fitter individual found until here represents the near optimum solution (Michalewicz, 1996).

There are several approaches to implement the genetic operators (crossover and mutation), and selection methods. The choice of these different approaches should be based on the problem at hand. Some may work well in a type of problem, but not as well in another problems. Therefore, to build a successful genetic algorithm it should be taken into account the operators and methods utilized. Literature have proposed several different genetic operators and selection methods (Srinivas, 1994).

6.2.2 Encoding

Encoding of candidate solutions is one of the most important problems when implementing a Genetic Algorithm. Encoding consists in creating a representation (chromosome) of the individuals in the population. A chromosome is a vector of genes, each gene represents a characteristic of the chromosome. Encoding depends on the type of problem, for example,

binary encoding is the most frequently used. In binary encoding the chromosome is represented by a vector of bits 0 and 1.

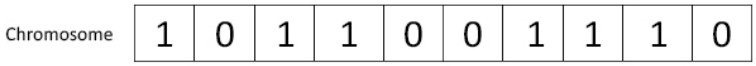

Figure 15 – Binary encoding, adapted from (Srinivas, 1994).

However, this encoding is not natural, and is not suitable to represent the candidate solutions for some problems. Therefore have emerged the encoding techniques (Srinivas, 1994): value encoding, tree encoding, among others.

Value encoding is especially useful in problems where the solutions are very complicated values or impossible to represent by a binary encoding. In a value encoding, a chromosome is a vector of values, which can be for example, real numbers, words, among others, in sum it could be anything related with the problem at hand (Srinivas, 1994).

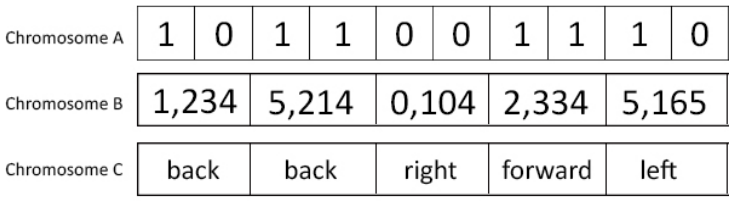

Figure 16 – Value encoding, adapted from (Srinivas, 1994).

Tree encoding (Srinivas, 1994) is mainly used in Genetic Programming algorithms, which evolve programs or expressions. In this technique each chromosome is a tree of objects related with the problem at hand, such as: functions, commands, among others.

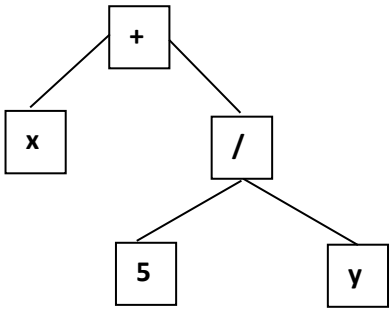

Figure 17 – Tree encoding, adapted from (Srinivas, 1994).

6.2.3 Selection

After evaluating the fitness of each individual in the population, a selection method is used to choose the individuals to submit to the reproduction process, in order to produce on or more offspring. Some selection strategies commonly used are (Abraham, 2005; Michalewicz, 1996):

Roulette wheel (fitness proportional selection), ranking based selection (linear and nonlinear), tournament selection, and elitism.

Roulette-wheel (Abraham, 2005; Michalewicz, 1996) selection is the simplest approach. It is considered a proportional selection, because the probability of an individual being select is proportional with its fitness-value. This technique simulates a roulette wheel, in which each individual is mapped to a slice of the wheel and its size is equal to its fitness. A random number is generated, and the individual of the slice spans that matches the random number is selected. The figure Figure 18 presents this technique, as can be seen the chromosome 1 has the higher probability of being selected.

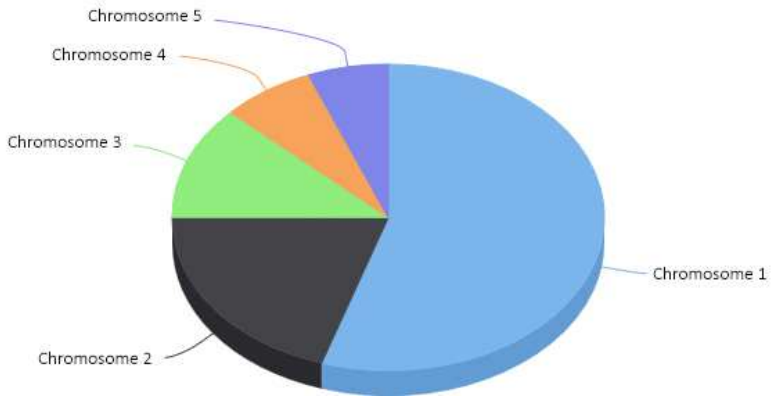

Figure 18 – Roulette-wheel selection, adapted from (Abraham, 2005).

In **ranking based selection** (Abraham, 2005; Michalewicz, 1996), the individuals are selected by their fitness value. They are sorted from the best to the worst, and the best will be selected.

Tournament selection consists in choosing randomly a number of chromosomes from the population, and the best individual is selected as a parent. This process is repeated until there is the desired number of individuals to reproduce.

6.2.4 Crossover

The genetic operator crossover has the goal of generating one or more offspring combining the genes from two chromosomes (individuals). Literature refer several techniques for that, being the most common (Abraham, 2005): single point crossover, two point crossover, k-point crossover, uniform crossover, intermediate crossover, global discrete crossover.

The simplest approach is the **single point crossover**, in which it is randomly selected a crossover point and all the genes before the point are copied from the first parent, and all the genes after the point is copied from the second parent, as can be seen in the example below.

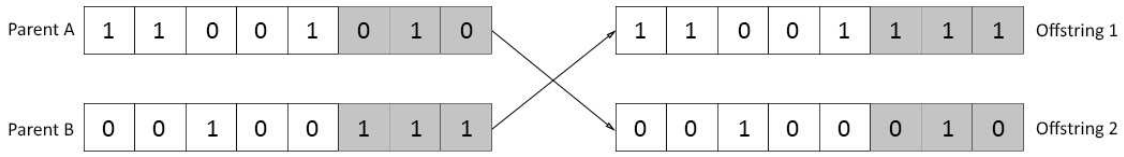

Figure 19 – Single point crossover.

In the **two point crossover** (Abraham, 2005), two points are randomly selected. The genes between the two crossover points are copied from the second parent and the remaining genes are maintained.

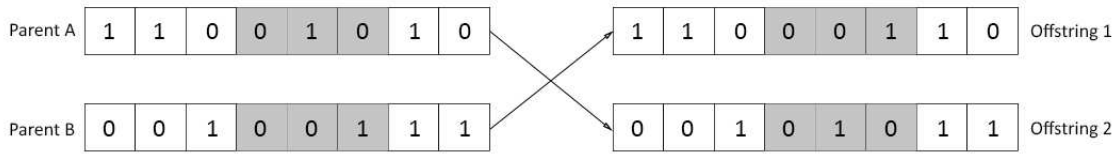

Figure 20 – Two point crossover.

In the **arithmetic crossover**, a new offspring is generated by applying some arithmetic operation.

1 1 0 0 1 0 1 1 + 1 1 0 1 1 1 1 1 = 1 1 0 0 1 0 0 1 (AND)

Figure 21 – Arithmetic crossover.

In the **uniform crossover**, bits are randomly copied from the first or the second parent.

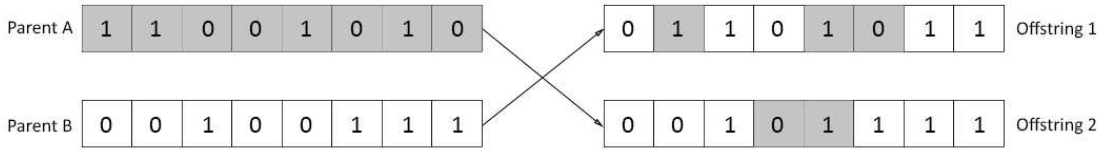

Figure 22 – Uniform crossover.

6.2.5 Mutation

The genetic operator mutation has the goal of creating diversity in a certain population. It randomly changes one or more genes of an offspring. Mutation depends on the encoding in the same way that crossover does. Literature has been describing several approaches to implement the mutation operator, being the most common: flip bit, swap and shift (Abraham, 2005).

The **flip bit** (Abraham, 2005) is the most simple mutation operator, which consists in randomly selecting a gene and invert its value (0 goes to 1 and vice versa).

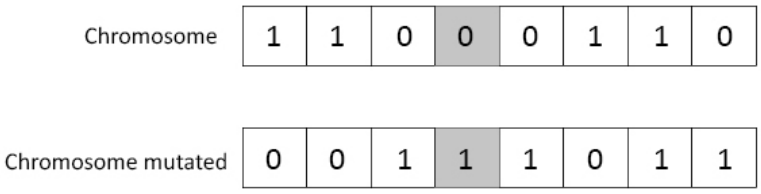

Figure 23 – Flip bit mutation.

The **swap** (Abraham, 2005) exchanges the position of two randomly selected genes.

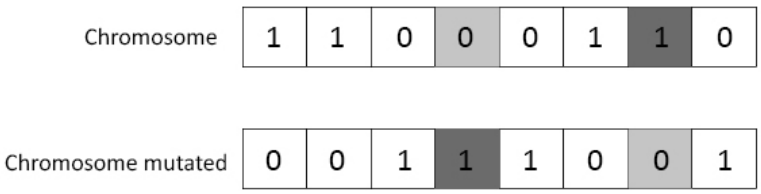

Figure 24 – Swap mutation.

The **shift** mutation operator shifts one gene to the right or to the left n positions.

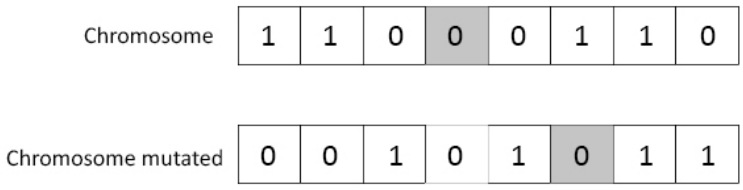

Figure 25 – Shift mutation.

6.3 Hyper-heuristics

This chapter overviews an approach to solve search and optimization problems, which have been widely addressed in literature, and is named as Hyper-Heuristic (HH) (Burke et al., 2003). The term hyper-heuristic was introduced as a high-level heuristic (Burke et al., 2003; Bruke et al., 2013). A heuristic can be defined as a technique that is specially designed to solve a certain problem. They can the categorized in several levels (Edmund et al., 2006): heuristics (low-level), meta-heuristics, and hyper-heuristics (high-level).

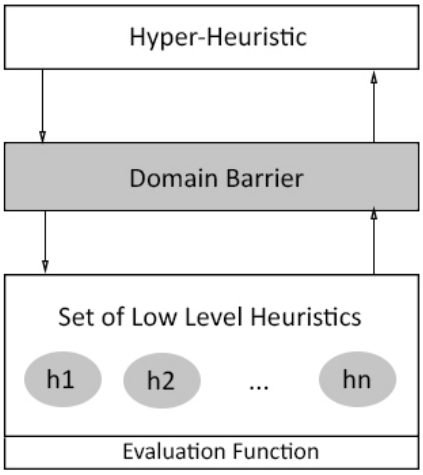

Figure 26 – Hyper-heuristic framework, adapted from (Burke et al., 2003).

(Burke et al., 2010) proposed the following definition of HH: "Hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational problems" (Ross, 2005). Therefore, a hyper-heuristic searches over a space of heuristics instead of a space solutions as it happens in the meta-heuristics. This search process is based on generation or selection of the right low level heuristics to efficiently solve the problem at hand, rather than solving the problem directly (Burke et al., 2003; Burke et al., 2013; Ozcan et al., 2013). This way, it can be achieved a near optimal solution using cheap and easy-to-implement low-level heuristics, instead of using an expensive computational approach (Cowling et al., 2002).

Knowledge-rich approaches can perform very well but are not easy to conceive and can have a higher computational cost (Burke et al., 2003). Therefore, there is a need to design methods that produce solutions of acceptable quality, but that are based on easy-to-implement low-level heuristics (Burke et al., 2013). Furthermore, in the majority of the real-world optimization problems there is no reasonable hope to find the very best solution to a certain problem. The search space can be extremely large and the use of exact methods can be unfeasibly expensive. Also, for real-world manufacturing organizations, in practice, it is not essential to find the optimal solution to a certain problem. They prefer good solutions that are found fairly quickly. These solutions should be "good enough" solutions, according to some minimum acceptance criteria (Burke et al., 2003; Ross, 2005). Therefore, in such cases it is common to use some kind of heuristic approach. It does not guarantees the optimal solution, but rather guarantees a good enough solution for the sake of computational speed.

Hyper-heuristics offers a possible way to efficiently solve this type of problems (Ross, 2005). They have been playing an important role in domains like: Artificial Intelligence and Operational Research. Also, they have been successfully applied to solve real-world computational search problems in areas such as: scheduling, data mining, bioinformatics, and stock cutting, among others (Burke et al., 2003; Bruke et al., 2013). The advantage of the use of a hyper-heuristic is that it leads to more general systems, which will be able to handle a

wide range of problem domains, instead of being customized to solve a particular problem or a set of problems as current meta-heuristics approaches do. This could facilitate the development of easy-to-implement methods that can operate on a range of related problems instead of one specific problem (Burke et al., 2003).

A hyper-heuristic is essentially a meta-heuristic that works at a higher level than the typical application of meta-heuristics to optimization problems. Therefore, a hyper-heuristic can be defined as meta-heuristic that manages a set of low-level heuristics (Burke et al., 2003). It searches for a good method to solve the problem at hand, instead of searching for a good solution. There are two main classes of hyper-heuristics commonly addressed in literature (Burke et al., 2005): methodologies to generate heuristics, and methodologies to select heuristics (Ozcan et al., 2013). In (Burke et al., 2010) can be found a classification of different approaches of hyper-heuristics, which are described in the following section. Some approaches produce constructive heuristics, which builds a solution to a problem. While other approaches aim to improve the quality of an initial heuristic. Furthermore, there is some approaches that produce a hybrid heuristic of these two (Ross, 2005).

6.3.1 Classification of Hyper-heuristics

The figure Figure 27 presents the classification of hyper-heuristic approaches proposed in (Burke et al., 2010). As can be seen in the figure, there are two main dimensions: the nature of the heuristics' search space, and the different feedback. In the nature of the heuristic search space, there are two distinct approaches (Burke et al., 2013): selection and generation. Selection hyper-heuristics can select or choose existing low-level heuristics to construct (constructive), or improve (perturbative) a solution to a certain problem. A generation hyper-heuristic can generate new low-level constructive or perturbative heuristics (Burke et al., 2013; Pillay, 2014). Perturbative methods consider complete candidate solutions and change them by modifying some solutions components, in order to turn them fitter to the problem at hand. On the other hand, constructive methods considers only partial candidate solutions, in which some solution components are missing and extends them (Burke et al., 2013).

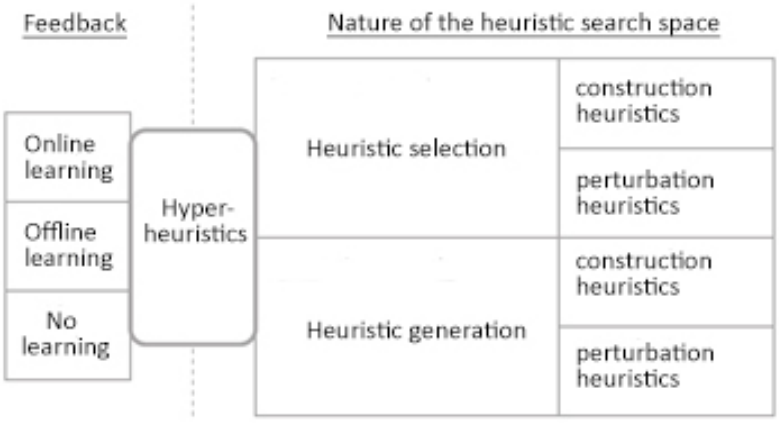

Figure 27 – Classification of Hyper-heuristics, adapted from (Burke et al., 2013).

A hyper-heuristic can also be considered a learning algorithm, if it uses some feedback from the searching process. According with the feedback, the learning process can be online or offline. The learning process consists in gathering knowledge from a set of training data, in order to be able of generalise solutions to new problems. The difference between online learning and offline learning, is that in online learning the algorithm also learns over time, by resolving new problem instance, in offline learning that does not happen, the knowledge that was gathered during the training is what it got over time. These are just some approaches commonly described in literature (Burke et al., 2013). However, many others exist, an example is the hybrid approaches that combine constructive with perturbation heuristics, or selection with generation heuristics (Burke et al., 2013).

A selection constructive hyper-heuristic builds a solution iteratively. They start with a partial or empty solution and they intelligently choose the constructive low-level heuristic to iteratively construct a solution. Therefore, a selection constructive heuristic receives as input the problem specific objective function and low-level construction heuristics, and to each current problem state selects the most suitable heuristic. This process is performed during several iterations, until a complete solution has been reached. Evolutionary algorithm hyper-heuristics have been successfully applied to domains like: educational timetabling, production scheduling, bin packing and cutting stock problems (Burke et al., 2013; Pillay, 2014).

A selection perturbative hyper-heuristic has the goal of improving a candidate solution to a certain problem, by automatically selecting and applying a low-level heuristic. Perturbative hyper-heuristics methodologies have been applied to a wide range of combinatorial optimization problems. The selection of perturbative heuristics can be performed by multi-point search or single-point search (Edmund et al., 2006). Multi-point selection perturbative Hyper-heuristics are population-based (processes multiple solutions). The hyper-heuristic employs a meta-heuristic (like a Genetic Algorithm, Ant Colonization, Particle Swarm Optimization, among others) to explore the space of low-level perturbation heuristics. The single-point selection perturbative hyper-heuristics have two major components (Edmund et al., 2006): a heuristic selection method and a move acceptance method. There are several approaches to heuristic selection method: random selection, roulette wheel selection, selection of the heuristic with the highest score, reinforcement learning, among others. Move acceptance methods can be deterministic or non-deterministic. Deterministic methods produce always the same result to the same input. Non-deterministic methods are stochastic methods, like for example: simulated annealing (Burke et al., 2013; Pillay, 2014).

Generation hyper-heuristics searches a space of heuristics already constructed from components, instead of searching a complete, pre-defined space of heuristics. This type of heuristics will output the new generated heuristic, which can be reused on new problem instances. An evolutionary algorithm widely used to implement generation hyper-heuristics is the Genetic Programming algorithm, which evolves a population of computer programs (Burke et al., 2013; Pillay, 2014).

6.4 Chapter Summary

In short this chapter introduced and overviewed techniques to solve search and optimisation problems, evolutionary computation (section 6.1) and hyper-heuristics (section 6.3). This chapter presented the biological motivation and fundamental aspects of evolutionary computation and its algorithms: genetic algorithm, evolution strategies, evolutionary programming, and genetic programing (Abraham, 2005). These are based on the principle of evolution inspired ty the Darwin' theory: survival of the fittest. Genetic algorithm was discussed in detail in section 6.2, which describes the entire working flow of a genetic algorithm and different approaches to implement genetic operators.

Hyper-heuristics plays a major role in search problems. Hyper-heuristics raise the level of generality at which optimisation systems can operate. In other words, hyper-heuristics lead to more general systems that are able to handle a wide range of problem domains, instead of being customized to solve a particular problem or a narrow set of problem domains, as meta-heuristics do (Burke et al., 2003). Section 6.3.1 of this chapter lists and describes a classification of hyper-heuristics. A hyper-heuristic is essentially a meta-heuristic that works at a higher level than the typical application of meta-heuristics to optimization problems. Therefore, a hyper-heuristic can be defined as meta-heuristic that manages a set of low-level heuristics (Burke et al., 2003). It searches for a good method to solve the problem at hand, instead of searching for a good solution.

7 Dynamic Integration Module Prototype

This master thesis was integrated on the development of R&D project ADSyS (Adaptive Decision Support System for Interactive Scheduling with MetaCognition and User Modeling Experience) (PTDC/EME-GIN/109956/2009), which is a multi-agent system to resolve scheduling problems subject to disturbances. This project was developed at the research group GECAD. The ADSyS project has the goal of plan and control decisions on the acquisition, utilization and allocation of manufacturing resources, in order to contribute to achieving goals and profit maximization of organizations at manufacturing environments.

Current real-world manufacturing systems have a dynamic nature, due to different disturbances that can occur on working conditions and requirements over time (a breakdown machine, urgent job arrival, job cancelation, and change in job priority, among others). Therefore, nowadays scheduling systems should have the ability to efficiently and effectively adapt and incorporate these dynamic events in the existing schedules, on a continuous basis (Madureira et al., 2014). This way, in this chapter is proposed a dynamic integration mechanism module based on selection constructive hyper-heuristic to allow the ADSyS scheduling system to decide autonomously the best match of integration mechanisms to incorporate new tasks in the current scheduling plan.

7.1 ADSyS System

Throughout this topic is analysed the existing ADSyS system in order to understand and contextualize the nature of the problem. The scheduling system ADSyS is a Multi-Agent System (MAS), which models a real-world manufacturing environment susceptible to disturbances. Like other scheduling systems, the ADSyS system has the goal of allocating limited resources to jobs under a number of constraints, in an efficient manner. This allocation of resources must be done in way so that the organization optimizes its objectives and achieves its goals. Therefore, the ADSyS system uses optimization techniques, namely Meta-

heuristics (i.e. tabu search, simulated annealing, ant colony optimization, particle swarm optimization, among others) to obtain near-optimal solutions of scheduling plans. In addition, due to the dynamic nature of current real-world manufacturing systems, the ADSyS system uses integration mechanisms to adapt to dynamic events that can occur over time, such as: arrival of new job, job cancellation, alteration of a job due date, among others). Throughout this chapter this will be addressed in more detail.

The ADSyS system is composed by three main modules (Madureira et al., 2014): the integrated interface module, the scheduling module, the user modelling module, and the dynamic adaptation module. Additionally, four interaction modules are considered: Task Editor, Machines Editor, Order Set Editor and Gantt Chart Editor, which are responsible for the input data, to the definition of a scheduling problem. Figure Figure 28 is present the ADSyS system global architecture and describe in more detail each module. The arrows in the figure represent the flow of information between the system components.

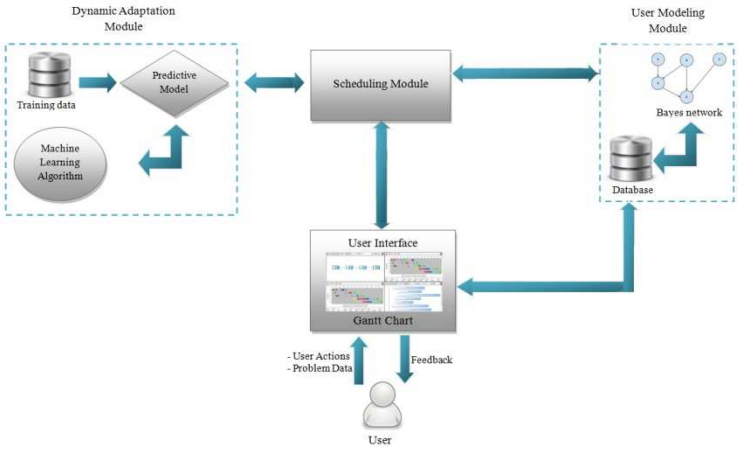

Figure 28 – ADSyS architecture, adapted from (Madureira et al., 2014).

- **Integrated interface module**

The integrated interface module allows the interaction between the user and the system. It allows the user the definition of a new scheduling problem, as well as, the visualization of the scheduling plan resulted. The user can dynamically interact with the scheduling plan, which is represented by a Gantt chart and a report summary. This module also has a multi-view feature, which consists in allowing the user to view and interact with different windows simultaneously. It provides the ADSyS system with a high degree of flexibility and information.

  - **Task Editor**

The task editor allows the user to create or edit tasks. Since a task is composed by one or more operations, the user should define the sequence in which the operations must be performed. The figure Figure 28 presents the task editor window, as it can be seen, the operations are represented by squares and the arrows define the precedence between them. For each operation the user should define an operation description, identify the machine where the operation should be executed, as well as, the processing time of the operation (green square in the figure Figure 29).

Through the tools panel (blue square in the figure Figure 29), the user can insert and remove operations and precedence to define a new sequence of operations. This panel also has useful options in order to improve the user interaction with the system, which are options to redo and undo performed actions, and options related with the selection mode of operations: select multiple, select all and deselect all.

To ensure that a task was correctly defined by the user, the system uses a real time validation mechanism. Each modification to a task is instantaneously validated and, in case of failure, the state object (red square in the figure) changes its appearance to a wrong symbol and a tooltip message is available, so that the user understands the cause of the error and how to correct it. This validation mechanism is also presented in the machines and order set editors.

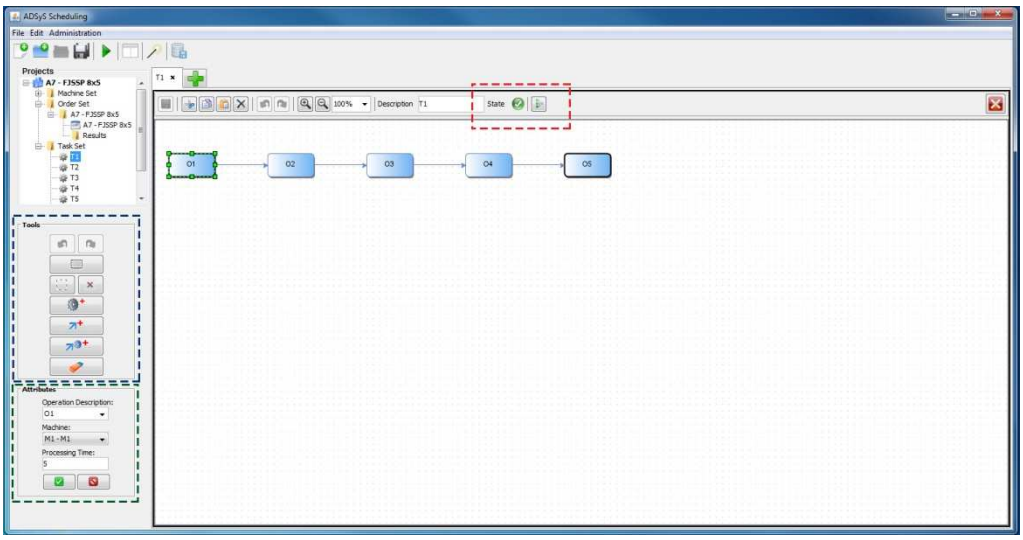

Figure 29 – ADSyS task editor.

- **Machines Editor**

The machine editor (figureFigure 30) allows the user to delineate the machines that are available for processing operations. This editor allows the user to create, remove or modify machines (blue square in the figure 30). The number of machines is not restricted and they can be inserted and not utilized later in the final plan. To each machine the user should define an ID and description.

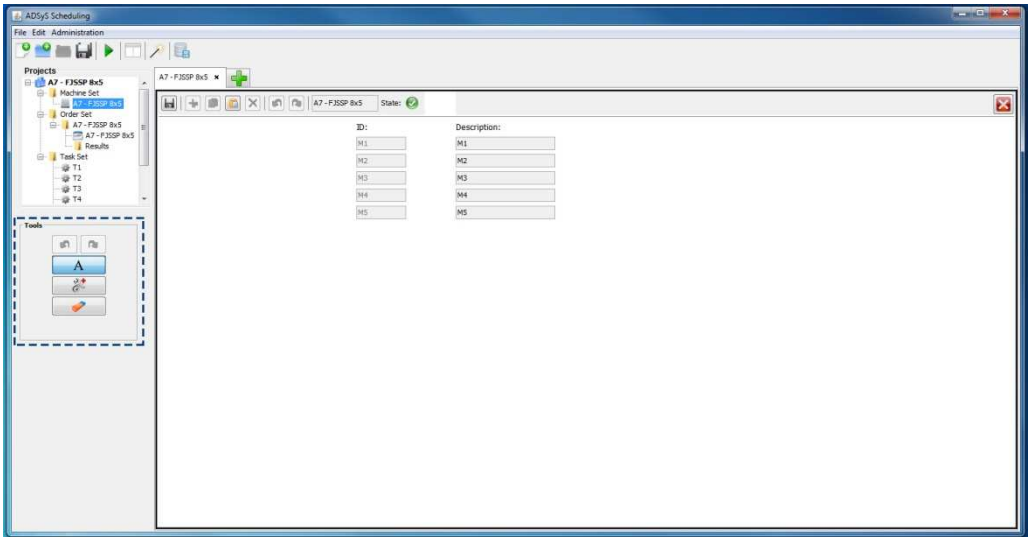

Figure 30 – ADSyS machines editor.

- **Order Set Editor**

The order set editor allows the definition and edition of a scheduling problem. A scheduling problem corresponds to a set of tasks to be executed during a given period of time. To each task the user should define the release date, the due date, the weight and the quantity.

The figure Figure 31 presents the ADSyS order set editor, as can be seen, the scheduling problem definition is represented by an ambient graphic. This ambient graphic has the goal of improving the user interaction with the system. Thus, the user can insert, remove, switch and modify the tasks by changing the attributes directly (green square in the figure Figure 31) or interacting with the graphic using a pointing device. There are two types of movements that can be applied to the tasks, using the pointing device: horizontal and vertical. When the user moves a task horizontally, the release and due dates are modified, according to where the task has been released. In vertical movements the user is capable of switching the position of one or many tasks simultaneously, by dragging and releasing a given task to the intended position. Other operation that switches the positions of the tasks are removing a task or inserting a new one, which can be done using the tools window (blue square in the figure Figure 31). The user can also resize a task to change the task duration and one of the dates (either due date or release date).

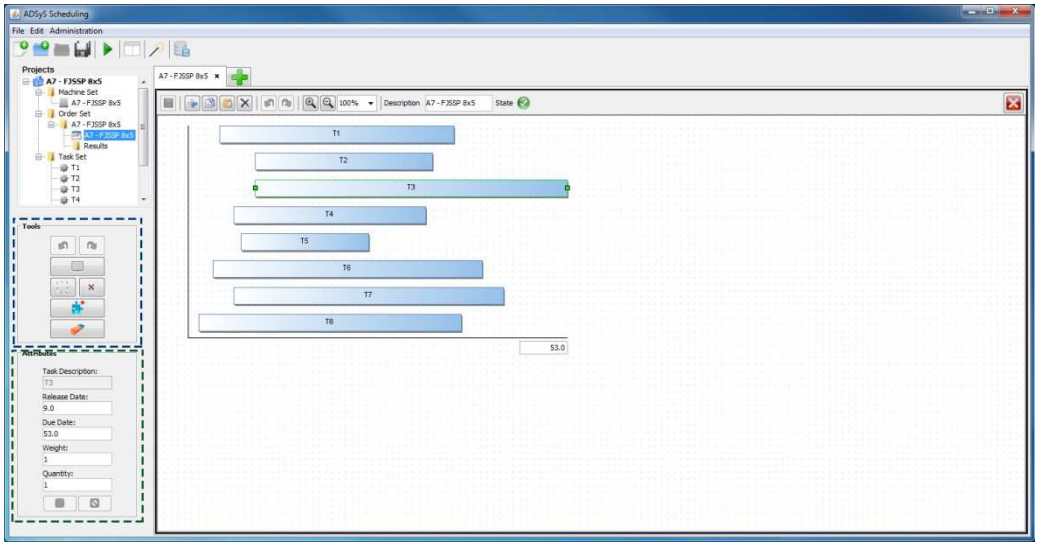

Figure 31 – ADSyS order set editor.

- **Gantt Chart Editor**

The scheduling plan resulted from the problem defined by the user is represented by a Gantt chart Figure 32. Similarly to the order set editor, the user can dynamically edit operations using the attributes window (blue square in the figure), or using a pointing device, through drag and release actions. The user can delay or anticipate an operation in a given machine by performing horizontal movements. When a movement is performed, the system does two validations: operation overlapping, where two operation are not able to be executed at the same time by one machine, and task constraints, where an operation is only executed after the task release date and once the corresponding (if any) preceding operation is finished.

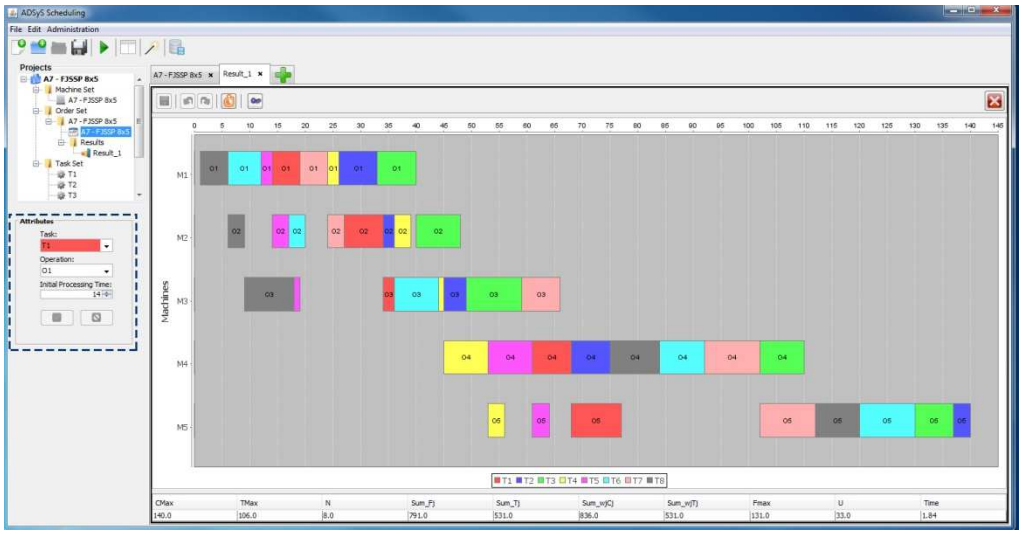

Figure 32 – ADSyS gantt chart editor.

- **Scheduling module**

The scheduling module is the scheduler itself, it is responsible for constructing a scheduling solution to the problem defined by the user, using different algorithms such as: tabu search, genetic algorithm, simulated annealing, ant colony optimization, particle swarm optimization, artificial bee colony and even memetic algorithm. Also, the scheduler is able to deal with dynamic events, such as: arrival of new tasks, cancelled tasks, tasks attributes modification, among others. To incorporate these events the scheduler uses different techniques: Earliest Due Date (EDD), Earliest Release Date (ERD), Greatest Priority First (GPF) and First In First Out (FIFO). These techniques are explained in more detail in the section 7.2 of this chapter.

- **User modeling module**

The user modeling module has the purpose of easing the learning curve of new users while boosting the productivity of expert users. It allows the system to analyse and characterize the user behaviour, in order to adapt itself to the user expertise level. It uses a Bayesian Network to mathematically analyse the user actions and categorize the level of expertise of the user. There are considered three levels: beginner, intermediate and expert (Madureira et al., 2014).

Therefore, the ADSyS system interface will adapt itself to the level of expertise of the user that is using the system. The main differences between each level that the interface provides are the quantity of tips, the automation and the interface. At a lower level the system will offer multiple tips and suggestions, ranging from tutorials to full explanations of the functionalities. A user without expertise will also have more automation from the system (e.g. filling complex forms with appropriate values). At the upper level, an expert user does not need the same amount of help and is able to decide the level of automation provided by the system. The prototype also offers a configuration option to allow the user to change its own level, assuring that each one has the optimal experience within the system (e.g. an expert user may want multiple tips with a lower level of detail) with a higher usability.

- **Dynamic adaptation module**

The dynamic adaptation module is responsible for updating the existing scheduling plan to the unexpected events that can arise in the system over time, in order to maintain the plan feasible and efficient. When a dynamic event happens, (for example: a new task arrives), the system automatically chooses one of the integration mechanisms to incorporate the new task in the current scheduling plan, based on the current state and previous information. Since it is impossible to anticipate every single problem, this module is capable of learning by experience. Therefore, the ADSyS system uses a supervised machine learning approach to serve this purpose. It uses a classification algorithm to automatically decide which integration mechanism is more accurate to use in a certain situation. This dynamic adaptation module will be described in detail in section 7.2, since it is the main goal of this thesis.

7.1.1 ADSyS Architecture

Figure Figure 28 – ADSyS architecture, adapted from (Madureira et al., 2014). described the global architecture of the ADSyS system, mentioning the different modules and its interconnections.

In this section, it is described the Multi-Agent System architecture of the ADSyS system. Due to the complexity and dimension of a scheduling problem, it is advantageous the use of a Multi-Agent architecture to model the scheduling environment. This MAS architecture is composed by four classes of agents, as can be seen in the figureFigure 33, which are: UI agent, Task agents, Resources agents, and Self*- agents.

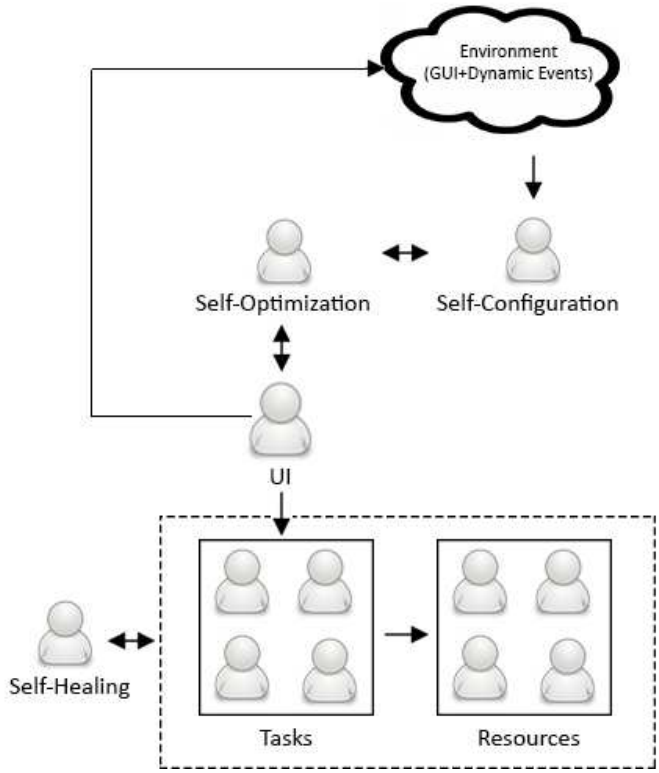

Figure 33 – ADSyS Multi-Agent System (Madureira et al., 2014).

- **UI Coordinator Agent**

The UI Agent provides a reliable communication between the user and environment. This agent is responsible for the definition of the graphical interface and for the dynamic generation of the necessary task and resource agents, according with the number of tasks and machines in the scheduling problem. Also, it assigns each task to the respective task agent. In the end of the process, this agent receives the solutions generated by the different resource agents and applies a repair mechanism in order to check the feasibility of the final scheduling plan.

- **Task Agents**

Task agents are responsible for processing the necessary information about the respective assigned task. They generate the earliest and latest processing times of the respective task and automatically assign each operation for the respective resource agent.

- **Resource Agents**

Each resource agent is responsible for scheduling the operations assigned to him, using a Meta-heuristic. In the end of the scheduling they communicate those solutions to the UI agent;

- **Self-* Agents**

In order to provide the system with autonomous behaviour, three self-* agents were incorporated:

  - **Self-Configuration Agent**

    The self-configuration agent allows a dynamic adaptation of the ADSyS system. This agent monitors, on a continuous basis, the system in order to detect changes in the current schedule plan. For example, when a new task arrives in the system it communicates with the dynamic adaptation module to select the most appropriate IM to use, according to the current scheduling problem;

  - **Self-Optimizing Agent**

    The self-optimization agent is responsible for the automatic parameterization of the Meta-heuristics, according with the scheduling problem;

  - **Self-Healing Agent**

    Finally, the self-healing agent is always monitoring the others agents, in order to keep them always running.

7.2 Dynamic Adaptation Module

Since current real-world manufacturing organizations have a dynamic nature, where unexpected events are constantly happening, it is fundamental that current scheduling systems have the ability to incorporate these dynamic events into the scheduling plan of a manufacturing organization in an efficient manner. As result, the scheduling plan that the manufacturing system is following will be maintained feasible, and the organization keeps clients satisfied and achieves its objectives (costs minimized and profit maximized). The dynamic adaptation module of the ADSyS system serves this purpose.

To incorporate a dynamic task into a scheduling plan already existing, the ADSyS system implements four different Integration Mechanisms (IM), which are (Madureira et al., 2014):

- **EDD (Earliest Due Date) –** The new task is integrated on the current schedule according to its due date. The operations of the new task are inserted before the operation with a greater due date.

- **ERD (Earliest Release Date) –** The new task is integrated on the current schedule according to its release date. The operations of the new task are inserted before the operation with a greater release date.
- **GPF (Greatest Priority First) –** The new task is integrated on the current schedule according to its priority. The operations of the new task are inserted before the operation with the lower priority.
- **FIFO (First In First Out) –** The new task is integrated at the end of the current schedule. The operations of the new task will be integrated on the current schedule according to its arrival date.

The primary goal of this module is to decide autonomously which IM is more suitable to integrate a new task in the scheduling plan at hand. However, it is not possible to find a reliable IM that will integrate in an efficient manner any dynamic task into any scheduling problem. In general, some IMs work well for particular problems, but not for all of them. Therefore, there is not an optimum IM, it may vary with the type of problem. In this context a problem can be characterized by the information related to the scheduling plan at hand and the information about the new task. As result, the ADSyS system should have a mechanism that dynamically chooses the most efficient IM to a certain problem. The approach already implemented in the ADSyS system that addresses this issue consists in the use of supervised machine learning techniques. In this approach a supervised algorithm learns over time by experience, in order to be able to predict the most suitable IM to integrate a new task into the scheduling problem at hand. Section 7.2.1 describes in detail the implementation of this approach.

However, the aim of this master thesis is to propose a new approach to address the problem of incorporating a dynamic task in the schedule plan at hand, in an efficient manner. The major difference between this approach and the supervised learning approach is that the last one uses a single IM to integrate the new task into plan. As resource agents are responsible for scheduling and rescheduling of the operations assigned to them, all of them use the same IM to perform the integration. However, the approach presented in this work will not choose just one IM, but a combination of them. In way that for example: the machine 1 can use the EDD, the machine 2 can use FIFO, the machine 3 can use EDD, and so on. With this approach it is believed that better results can be achieved. Therefore, the goal of this approach is to determine which ordering of IMs exhibits the best results for a pre-established criterion. The criterion that will be used is the maximum time of completion of the plan, also called C_{max}. Thus, the sequence of IMs that achieves the minimum completion time is the optimal solution that the system should use to perform the integration. As this can be considered an optimization problem, the work in this thesis will validate the contribution of areas, such as: Aml and evolutionary computing on supporting the search for the near optimal solution, from adaptive perspective.

The problem of selecting an effective, good enough, or best solution arises in a wide variety of situations. Literature refers, that evolutionary algorithms like genetic algorithms, have been successfully applied to the resolution of optimization problems (Srinivas, 1994). A GA is a

robust intelligent optimization technique that heuristically searches for the near optimal solution into a large solution space, by applying evolutionary techniques based on the Darwin's theory of evolution. Unlike the supervised learning approach, this approach does not derivate information from previous experiences. It maintains a population of possible solutions, in which each solution consists in a combination of IMs, and over several generations (iterations) discards the worst solutions to converge to a near optimal solution, by applying genetic operators and an elitist selection.

Furthermore, according with (Pillay, 2014) this problem can also be categorized as a hyper-heuristic problem, because hyper-heuristics can be defined as a search method for selecting or generating low-level heuristics, instead of searching directly a solution space as meta-heuristics do. The IMs used in this work can be considered constructive low-level heuristics, since they are used to create a solution to a problem, in other words, they integrate the dynamic task in the current scheduling plan, using some criterion of decision.

Thus, the goal of this thesis is validate the potential of a selection constructive GA hyper-heuristic in finding the near optimal combination of IMs to integrate dynamic tasks into the scheduling plan at hand. Furthermore, in the chapter 8 is performed a comparison between this proposed approach and the supervised learning approach, in order to validate the efficiency of this approach in the optimization problem resolution. In the section 7.2.2 is presented in detail the implementation of this approach.

7.2.1 Supervised Learning Approach

The dynamic adaptation module already existent in the ADSyS system serves the purpose of integrate a dynamic task into the current scheduling plan, by implementing an intelligent environment using supervised machine learning techniques to autonomously and intelligently choose the integration mechanism that should be used in the problem at hand. Therefore, this module deals with the decision making problem of selecting the most accurate mechanism to incorporate new tasks in a certain problem in order to adapt a current schedule plan to external perturbations.

The figure Figure 34 presents the supervised machine learning approach implemented in the dynamic adaptation module. This approach has three major components: a predictive model, a database, and a supervised machine learning algorithm. The database contains the data used to train and test the learning system, which consists in a set of scheduling problems, each one labelled with the IM that presented the best results in integrating a new task. This data was acquired through observation of the performance of each IM integrating a new task into several different scheduling problems. The training data was used to train a supervised machine learning algorithm, namely a decision tree algorithm, and as result create a supervised classifier. This classifier has the responsibility of choosing the best IM to integrate a new task in the current scheduling plan.

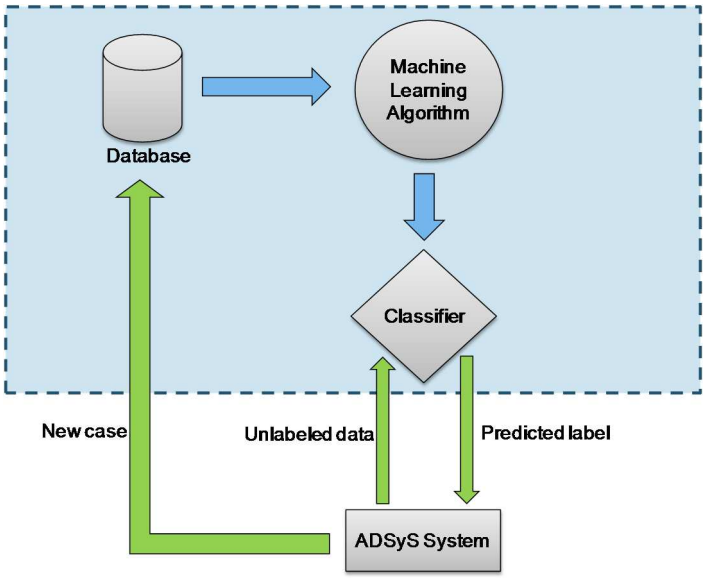

Figure 34 – Dynamic adaptation module architecture, adapted from (Madureira et al., 2014).

The supervised decision tree algorithm was chosen based on the obtained results of the previous work presented in (Madureira et al., 2014). The goal of this work was to identify the algorithm that had the better performance in the attempt to classify new tasks into one of four possible classes (EDD, ERD, GPF or FIFO), which correspond to the four different IMs. Therefore, it was performed several experiments, testing five different supervised algorithms, which are: Naive Bayes, decision trees, random forests, neural networks and support vector machines. In order to guarantee statistical significance there were performed 30 repetitions of each experiment. Also, the parameters of the algorithms were tuned in order to obtain the best possible results. The results of this experiment concluded that the decision tree algorithm was the one that presented the best results for this kind of problem. Furthermore, this is a simple algorithm, commonly used in classification problems and has been demonstrating success in many application areas, which was an additional factor in its selection.

The working flow presented in the figure Figure 34 – Dynamic adaptation module architecture, adapted from (Madureira et al., 2014). is simple. When a new task arrives in the ADSyS system is sent to the decision tree based classifier the information related with the new task and with the scheduling problem in which it will be integrated, the table 5 presents the complete list of information sent to the classifier.

The classifier will return the predicted integration mechanism (EDD, ERD, GPF or FIFO), and the system will use it to integrate the operations of the new task. However, each classification is subjective, and sometimes the best option for the system is not the best for the user. Therefore, the ADSyS system has in consideration that the user should always have the final decision, about what happens in the system. This way, before the system integrates the new task using the IM predicted by the classifier, it asks the user if he agrees with the IM that has been chosen or if he wants choose a different one.

Table 5 – Information sent to the classifier, adapted from (Madureira et al., 2014).

Attribute	Description
Machines	Number of machines in the scheduling problem.
Tasks	Number of manufacturing tasks to schedule.
Time event	Instant of the introduction of a new task.
C_{max}	Makespan-Total processing time.
F_{max}	Maximum flow time.
L_{max}	Maximum lateness.
WT	Total weighted tardiness.
Comp time	Computational time spent in creating the scheduling.

This user-centred approach has the goal of improving the user experience with the system, in order to make the ADSyS system more usable. More information about system usability was addressed in chapter 0. Furthermore, as the system should have the capability of learning through experience on a continuous basis, each time that it makes a different prediction, it adds to the database the new case learned, increasing this way the knowledge of the system. As result, the system self-improves and the classification process becomes more accurate.

7.2.2 Selection Constructive Hyper-heuristic Approach Proposed

Hyper-heuristics is a search method for selecting or generating the most suitable low-level heuristics to solve an optimization problem (Pillay, 2014). These low-level heuristics can be constructive or perturbative. The integration mechanisms implemented in the ADSyS system can be categorized as constructive heuristics, since they are used to create a solution to a problem, in other words, they are used to solve the problem of integrating a dynamic task in a current scheduling plan.

The approach proposed in this master thesis consists in employing an evolutionary algorithm as selection constructive hyper-heuristic, according with (Pillay, 2014). This HH should automatically find the near optimal combination of IMs to integrate a new task in a scheduling plan. The optimal solution is the one that will perform the integration in the most efficient way as possible. The evolutionary algorithm chosen was the genetic algorithm, since it has been successfully applied to the resolution of optimization problems (Pillay, 2014). Therefore, the genetic algorithm hyper-heuristic will explore the space of low-level construction heuristics (mechanisms: EDD, ERD, FIFO, and GPF) to discover the near-optimal combination of simple heuristics. The output of this HH will be the most suitable sequence of low-level heuristics to solve the problem at hand. This sequence determines the IM that each resource agent should use. Since the combinations are dependent on the number of machines of the scheduling problem at hand, the search space upon the HH will be working can be rather large.

The figureFigure 35, presents the block diagram of the proposed approach. The overall aim is to select the most suitable set of low level heuristics to solve the problem of integrating a new task in a scheduling plan. The high level heuristic (hyper-heuristic) operates on a set of low level heuristics. At the high level, the HH in each generation selects the most suitable sequence of low-level heuristics to a given problem. Whilst, the low-level heuristics are used to indirectly construct the solution.

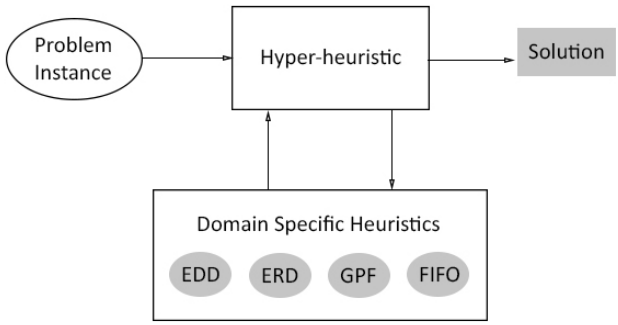

Figure 35 – Model of the proposed Hyper-heuristic.

The genetic algorithm hyper-heuristic (figure Figure 36) implemented starts with an initial set of random candidate solutions. Each candidate solution is a combination of single heuristics. The HH iteratively will evolve the candidate solutions until the stopping criterion has been met. In each iteration the candidate solutions are applied to solve the problem at hand, in order to assign a fitness value to each candidate. This value indicates how close a candidate solution is to an ideal solution. The progression of the candidate solutions is performed with elitism. The best solutions will be selected to reproduce and the worst are discarded. When the stopping criterion is met the best solution found will be used to make the effective integration of the dynamic task, applying to each resource agent the matched IM.

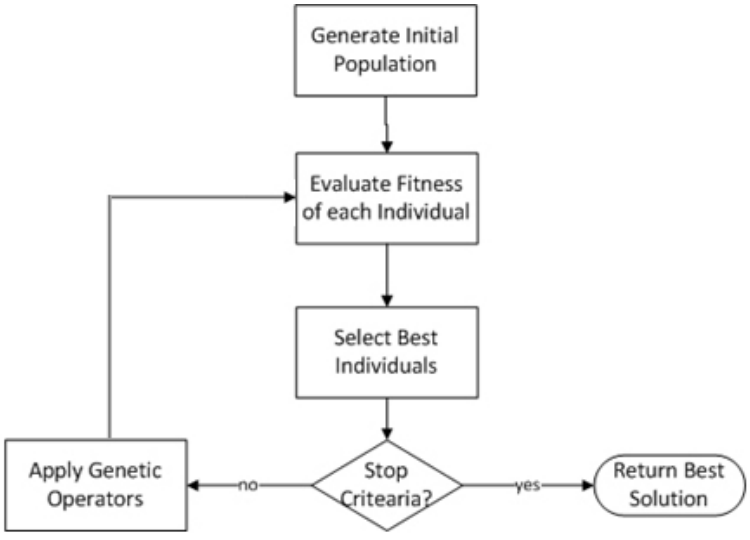

Figure 36 – Genetic Algorithm process flowchart.

As the solution space can be rather large, and the quality of each candidate solution must be evaluated by applying it to solve the problem at hand, this process demands of high computational effort. Therefore the appearance of two contradictory forces that must be balanced. On the one hand, the search process should not take too long, which implies relying on small instances and adopting parameters that minimize the computational overhead (e.g. population size, number of iterations). On the other hand, the adoption of excessively simple conditions can compromise the results. Therefore, in this thesis it is also performed an empirical study to determine the proper values for the different parameters used in the GA such as the population size, the number of generations, the crossover probability, and the mutation probability.

- **Coding to Strings**

In a GA it is necessary to find a suitable representation for each candidate solution in a population. Therefore, in this work each candidate solution (chromosome) is coded as a string of variable length. The length of the chromosome depends on the number of machines (resources) presented in the scheduling problem at hand.

An example of a possible chromosome is represented in the figureFigure 37. The chromosome comprises a set of low-level construction heuristics, which allows the integration of a new task.

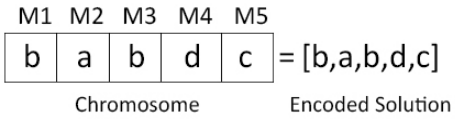

Figure 37 – Chromosome representation.

Each element of the chromosome represents the IM that should be applied to the corresponding machine. Each IM is represented by a letter between [a-d], in which the letter:

- ***a*** corresponds to the Earliest Due Date IM;
- ***b*** corresponds to the Earliest Release Date IM;
- ***c*** corresponds to the Greatest Priority First IM;
- ***d*** corresponds to the First In First Out IM.

Therefore, in the example babdc presented in the figure Figure 37, the machine M1 should use the ERD integration mechanism, the machine M2 the EDD, the machine M3 the ERD, the machine M4 the FIFO, and so on.

- **Initial Population Generation**

The initial population is composed by a predefined number of randomly generated chromosomes. As each chromosome is a set of low-level construction heuristics, each heuristic is randomly chosen from the four IMs (EDD, ERD, FIFO, and GPF). As each permutation of the four IMs can be a candidate to problem solution, it can be concluded that it is a problem with a very large search space. Also, it is worth noting that in the initial population generation replicated solutions are not allowed. When a candidate solution is

generated is verified if it already exists in the pool of the already generated random solutions. If the generated solution already exists, it is discarded and generated a new one.

- **Fitness Evaluation**

In each generation (iteration) of the GA, the fitness function is applied to evaluate the quality of each solution in the population. The fitness function associates to each candidate solution a fitness value, which represents how close is a solution to the satisfactory solution. In this work, the fitness value of a certain candidate solution is calculated by using the solution to integrate the new task in the scheduling plan at hand, and calculating the makespan of the resultant plan. Therefore, the fitness measure of the solution is the makespan, also called C_{max}, which is the time of completion of the scheduling plan, which the lower the better. The C_{max} is obtained by searching the operation with the latest conclusion time.

- **Selection of Parents**

The solutions that are submitted to the genetic operators (crossover and mutation) are chosen using the tournament selection technique. This technique randomly chooses a group of individuals, called the tournament, from the population and the fittest individual is selected as a parent. This way the poor solutions are removed and the best solution will remain in the population.

- **Genetic Operators**

Genetic operators drive the evolutionary process of a population in GA. These operators have the aim of causing population modification, in order to create new generations of individuals. According with the Darwinian' theory, the quality of the solutions in each generation is better than the previous. The GA implemented uses two genetic operators: crossover and mutation.

  - **Crossover**

The crossover operator aims to create offsprings fitter than their parents, which can probably happen if the offsprings inherit the best characteristics of them. The crossover technique selects a pair of candidates, and it recombines them with probability P_c to form two new candidates. In this work it is used 2-point crossover operator. This operator is exemplified in the figure 38, and consists in:

1. Generate randomly two crossover points;
2. Each descendant is generated inheriting, the other ascending genes that are on the segment delimitated by the two points;
3. The order of the remaining genes is the same as the direct ancestor.

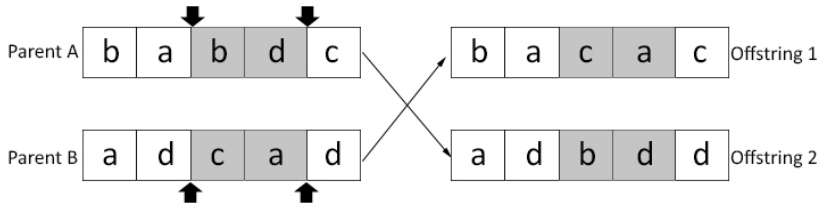

Figure 38 – 2-point crossover operator.

- **Mutation**

The mutation operator aims to lead the search to get out of a local optimum. This way, it adds diversity to the population at hand. This algorithm selects a candidate solution and applies mutation under the probability P_m. For mutation, a single low-level heuristic (gene) in a chromosome is randomly selected and replaced by another randomly chosen heuristic. An example of the mutation operator is shown in figure below. In the example, the chromosome *babdc* is a parent and *d* is the randomly selected heuristic, and *a* is the randomly selected heuristic chosen to replace *d*, resulting in the chromosome is *babac*.

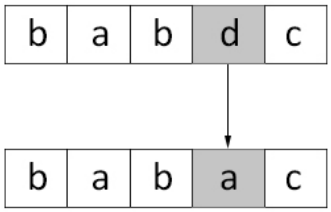

Figure 39 – Mutation operator.

7.3 Chapter Summary

In this chapter the ADSyS system was presented, which consists in a multi agent system with the goal of planning and controlling decisions on the acquisition, utilization and allocation of manufacturing resources under a number of constraints. As result, it can help current manufacturing organizations achieve their goals and maximization of their profit. However, current real-world manufacturing systems have a dynamic nature. Disturbances on working conditions and requirements often arise unexpectedly (new tasks arrives, others are cancelled, due dates change, among others). Therefore, scheduling systems should in an efficiently and effectively way adapt and incorporate these dynamic events into their current scheduling plans. However, it is not possible to find a reliable integration mechanism that will integrate in an efficient manner any dynamic task into any scheduling problem. In general, some integration mechanisms work well for particular problems, but not for all of them. Thus, a supervised machine learning approach was previously implemented in the ADSyS system, in order to serve this purpose. It uses a classification algorithm that automatically decides which integration mechanism is more suitable to integrate a certain task into a certain scheduling problem. This approach was described in the section 7.2.1, as well as, the performed experiments to find the most suitable supervised algorithm to the resolution of this problem.

The aim of this master thesis consists in proposing a new approach to address the problem of dynamically adapting the current scheduler to disturbances, in an efficient manner. The problem of selecting an effective solution arises in a wide variety of situations. Literature refers, that evolutionary algorithms have been successfully applied to the resolution of optimization problems. Therefore, the proposed approach of this thesis consisted in an implementation of an evolutionary algorithm namely the selection constructive hyper-heuristic. The chosen evolutionary algorithm was the genetic algorithm, since it is a robust

optimization technique. The major difference between this approach and the supervised learning approach is that the last one uses a single integration mechanism to integrate the new task into the entire plan. This approach uses a combination of integration mechanisms, in which each resource in the scheduling problem uses the most suitable integration mechanism, according with its characteristics. The hyper-heuristic searches for the near optimal combination of low-level heuristics (integration mechanisms) into a rather large search space. The optimal solution is the one that performs the adaptation of the scheduling plan in the most efficient way as possible. The criterion to decide which an optimal solution is, was the maximum time of completion of the plan (C_{max}). Section 7.2.2 describes the implementation of this proposed approach. The entire working flow of the evolutionary algorithm is explained, as well as, the different parameters used. With this proposal is believed that good quality solutions can be achieved when compared against the supervised learning approach, without the need of derivate information. In the next chapter is performed a computational study, in order to validate the performance of the proposed approach when compared with the previous approach.

8 Computational Study

To ensure that performance of the proposed selection constructive hyper-heuristic is as efficient as was expected, there is a need to test it. This chapter presents the experiments carried out during the development of this master thesis and the results obtained. These results are compared against those obtained by the supervised machine learning approach. In order to analyze and evaluate the applicability of the proposed selection constructive hyper-heuristic approach in the attempt of choose the near optimal solution to integrate a new dynamic task in a current scheduling plan, when compared it with the previous approach.

8.1 Experimental Setup

This section describes the experimental setup process of the performed experiments, which includes:

- Definition of a set of scheduling problems to test;
- The description and preparation of the adopted parameters;
- Definition of the performance metrics to evaluate behaviour of the new approach, against the supervised learning approach.

Considering that academic benchmark problems are an effective evaluation framework, and multiple authors and diverse application areas have used them over the years, the performance of the proposed approach, the evolutionary algorithm described in the chapter 7.2.2 was tested on 31 benchmark instances of Job-Shop Scheduling Problem (JSSP) of different sizes. The implementation of the entire ADSyS system, including the dynamic adaptation module addressed in this master thesis, was implemented using the programming language Java. The computational tests were carried out on a PC with the following specifications Intel Xeon W3565 at 3.20 GHz.

To each scheduling problem instance the genetic algorithm was iterated 75 times, in other words it uses 75 generations to converge to a near-optimal solution, and each generation maintains 25 candidate solutions. At each iteration of genetic algorithm process, the algorithm tries to replace candidate solutions with better ones. The probability of crossover used is equal to 0.7 (Pc = 0.7) and 0.01 is suitable value for probability of mutation (Pm = 0.01). These parameters were discovered by trial and error, table 6 lists them. The problem instance consists in a scheduling problem and the dynamic task that needs to be integrated in that problem instance. The search for the near optimum solution was carried out by the maximum completion time criteria (makespan). Therefore, the found solution that has the lower (C_{max}) value is considered a good solution.

Table 6 – Parameters of the genetic algorithm implemented.

Parameter	**Value**
Population Size	25
Number of Generations	75
Mutation Rate	0.01
Crossover Rate	0.7

As the main goal of this experiment is to compare the efficiency of the hyper-heuristic approach in relation to the supervised learning one, in integrating a new task in a current scheduler plan. The results were collected by trying to integrate the same dynamic task into the same scheduling problem instance using the two approaches, in order to guarantee statistical significance. After the integration of a dynamic task in a current scheduling plan, it can stay substantially the same or drastically different from the original plan after the integration. Therefore, in order to study and analyse the impact of dynamic events in a scheduling plan it was used the performance metrics described in the section 5.2.3 of this document, which are: utility, stability and robustness. Utility is the difference between the C_{max} before and after the integration of the task into the current scheduling plan. Stability is the sum of the absolute difference between the completion times of the scheduling plan before and after the integration of the new task. Finally, robustness of the final plan combines the maximization of the efficiency utility and the minimization of the deviation stability from the scheduling plan before integrating the new task. Robustness represents the insensitivity of scheduling performance to disturbances, which is a desirable feature in any scheduling system. The comparison of these two approaches is made based on this measures, section 8.2 demonstrate the achieved results. Furthermore, is noteworthy that each scheduling plan was generated using the meta-heuristic simulated annealing with the parameters listed in the table 7. To compare these two approaches it is important that all parameters and information used in each approach are the same, in order to guarantee statistical significance.

Table 7 – Parameter of simulated annealing.

Parameter	Value
Initial Temperature	15
Alpha	80
Number of Iterations ‘k’	30
Stopping Criteria	200

8.2 Computational Results

This section presents the obtained computational results in integrating a dynamic task in 31 benchmark instances of Job-Shop Scheduling Problem. To assess whether there is significant difference between the performance of hyper-heuristic and the supervised learning based approach it was used the t_student test for paired samples. The t_student is a statistical hypothesis test in which the test statistic follows a Student's t distribution. It can be used to determine if two sets of data are significantly different from each other. The assumption of normality was validated based on the central limit theorem (n>30).

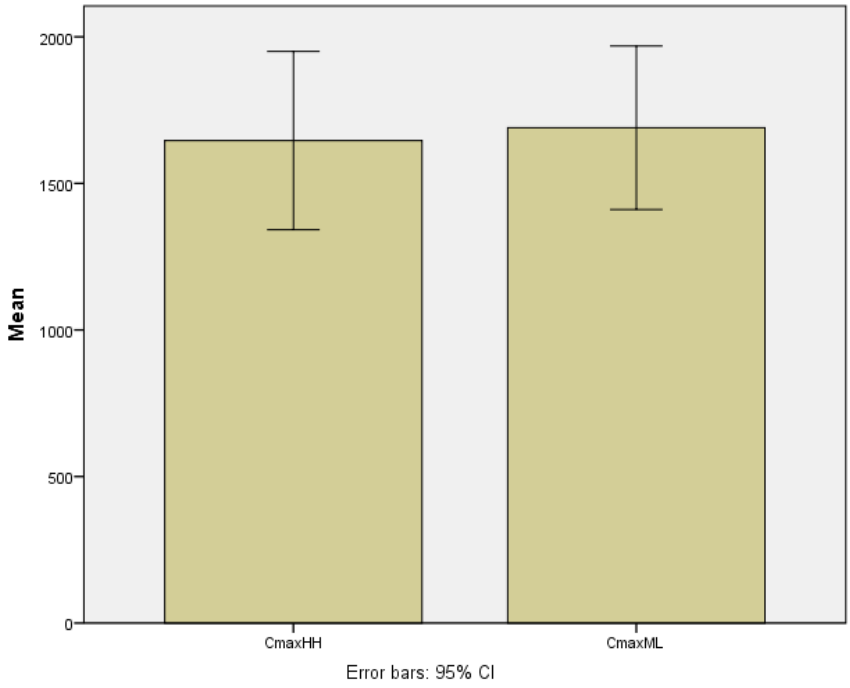

Figure 40 – Difference between the makespan of the two approaches.

As can be seen in the figure Figure 40, the mean of the makespan (C_{max}) obtained in the two approaches are very similar. The lower the C_{max} value the better. Therefore, can be concluded that the proposed hyper-heuristic approach has some advantages when compared with the machine learning approach.

Table 8 – T-student results for paired samples for makespan minimization.

	Paired Differences					t	df	Sig. (2-tailed)
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
				Lower	Upper			
HH - ML	-43,677	454,447	81,621	-210,370	123,015	-,535	30	,597

Table 8 shows the result of the Student's t-test. It is not possible to assume equal means the Student's t-test has a p-value of 0,2985 (0,597/2). The null hypothesis H0, that considers the non-existence of difference on hyper-heuristic and machine learning performances was retained with 95% of confidence level (p=0,2985 >α=0,05). This permits to conclude, with 95% of confidence, that performance of proposed hyper-heuristic is similar to machine learning based approach when analyzed *makespan* of rescheduled plans.

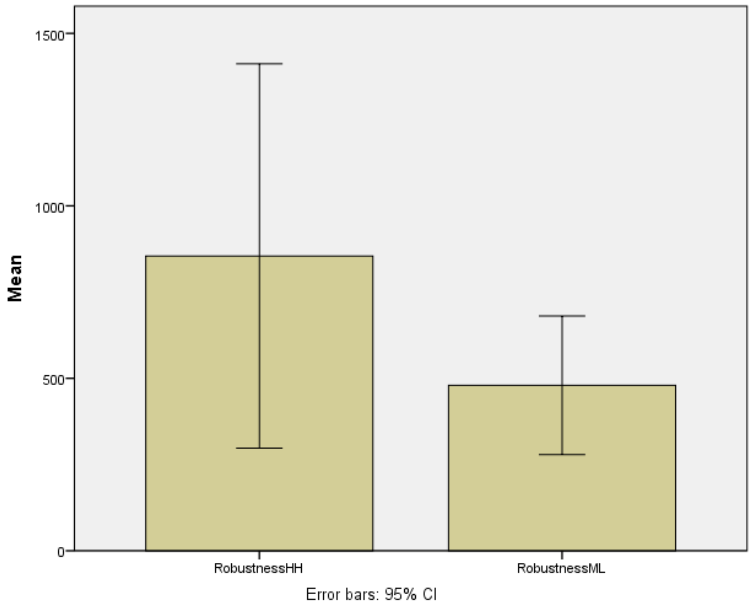

Figure 41 – Analyse of the robustness between the two approaches.

The figure Figure 41 compares the robustness results between the two approaches. From the analysis of the bar graph (figure Figure 41), it is possible to observe evidences of the advantage of using a hyper-heuristic approach when compared to the machine learning based approach, in the maximization of the robustness.

Table 9 – T-students results for paired samples for robustness maximization.

	Paired Differences					t	df	Sig. (2-tailed)
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
				Lower	Upper			
HH vs. ML	374,919	1585,091	284,691	-206,497	956,335	1,317	30	,198

Table 9 shows the result of the Student's t-test. It is not possible to assume equal means the Student's t-test has a p-value of 0,099 (0,198/2). The null hypothesis H0, that considers the non-existence of difference on hyper-heuristic and machine learning performances was retained with 95% of confidence level (p=0,099 >α=0,05). This permits to conclude, with 95% of confidence, that performance of proposed hyper-heuristic is similar to machine learning based approach when analyzed *robustness* of rescheduled plans.

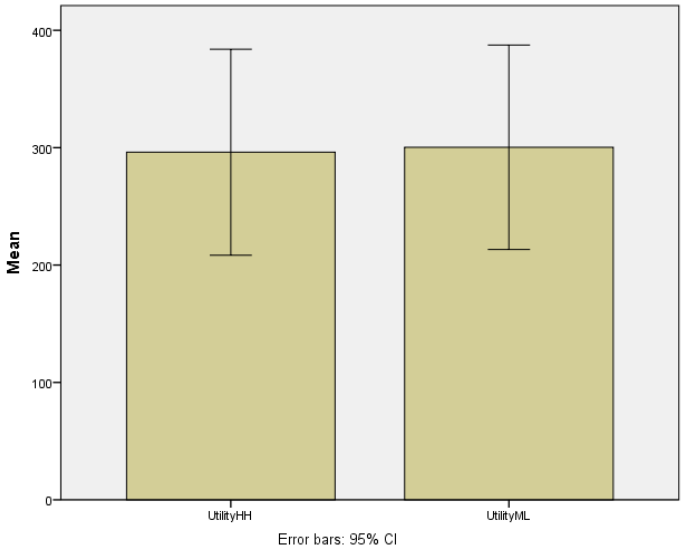

Figure 42 – Analyse of the utility between the two approaches.

The figure Figure 42 compares the utility results between the two approaches. From the analysis of the bar graph (figure Figure 42), it is possible to observe evidences of the advantage of using a hyper-heuristic approach when compared to the machine learning based approach, in the minimization of the utility.

Table 10 – T-student results for paired samples for utility maximization.

	Paired Differences					t	Df	Sig. (2-tailed)
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
				Lower	Upper			
HH vs. ML	-4,226	128,227	23,030	-51,260	42,808	-,183	30	,856

Table 10 shows the result of the Student's t-test. It is not possible to assume equal differences means the Student's t-test has a p-value of 0,428 (0,856/2). The null hypothesis H0, that considers the non-existence of difference on hyper-heuristic and machine learning performances was retained with 95% of confidence level (p=0,428 >α=0,05). This permits to conclude, with 95% of confidence, that performance of proposed hyper-heuristic is similar to machine learning based approach when analyzed utility of rescheduled plans.

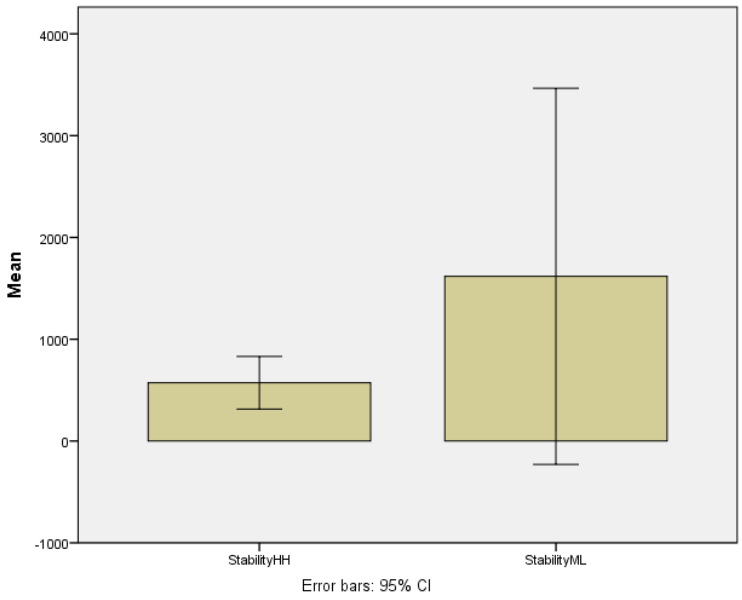

Figure 43 – Analyse of the Stability between the two approaches.

The figure Figure 43 compares the stability results between the two approaches. From the analysis of the bar graph (figure Figure 43), it is possible to observe evidences of the advantage of using a hyper-heuristic approach when compared to the machine learning based approach, in the minimization of the stability.

Table 11 – T-student results for paired samples for stability maximization.

	Paired Differences					t	df	Sig. (2-tailed)
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
				Lower	Upper			
HH vs. ML	-237,081	1271,339	228,339	-703,412	229,250	-1,038	30	,307

Table 11 shows the result of the Student's t-test. It is not possible to assume equal means the Student's t-test has a p-value of 0,1535 (0,307/2). The null hypothesis H_0, that considers the non-existence of difference on Hyper-Heuristic and Machine-Learning performances was retained with 95% of confidence level ((30) =-1,038; p=0,1535 >α=0,05). This permits to conclude, with 5% of significance level, that performance of proposed hyper-heuristic is similar to machine-learning based approach when analyzed *stability* of rescheduled plans.

In short, it is possible to conclude about advantages of proposed hyper-heuristic when analyzed means, although no statistical significance evidence has been proved. This means that we can evolve and refine it for real world scheduling problems. Additionally, it is possible to conclude that the proposed approach is more flexible since it proactively decides which integration mechanism will be used on each single machine of a scheduling problem. This way each machine will use the integration mechanism most suitable according with its characteristics.

8.3 Chapter Summary

This chapter had the goal of describing the evaluation process of the proposed approach. It described the experimental settings of the performed tests, indicating the types of problems used, as well as the parameters used. The two approaches were evaluated by integrating a dynamic task into 31 different scheduling problems. The results obtained from these experiments were analysed using the performance metrics described in the section 5.2.3 of this master thesis. The section 8.2 presents the conclusions drawn from this experimental process.

Looking at the results of the experiments performed, it can be concluded that the results obtained with proposed hyper-heuristic approach and with the supervised learning approach are very similar. Therefore, it can be considered that the proposed approach to integrate dynamic events in a current scheduling plan is efficient, at least with respect to the number of experiments performed. The hyper-heuristic approach is able to obtain good quality solutions when compared against the supervised learning approach. Therefore, it can be concluded that the use of different integration mechanisms to different machines can an asset to dynamic integration of dynamic tasks, since it is chosen the most suitable integration mechanism for a specific machine instead of the entire plan.

9 Conclusions and Future Work

Current manufacturing scheduling systems still have limitations in real-world environments. Since disturbances on working conditions occur on a continuous basis, it is important that current scheduling systems have the ability to automatically deal with these disturbances, maintaining the plan as feasible as possible. Therefore, the major goal of this dissertation was to address the problem of incorporating new tasks into the current schedule of a manufacturing organization, by proposing a selective constructive hyper-heuristic based on evolutionary algorithms, which applied concepts of the theory of evolution to solve complex computational problems.

This chapter resumes its most relevant contributions and highlights some lines of future work.

9.1 Main Contributions

Due to the dynamic nature of nowadays manufacturing organizations, dynamic adaptation and optimization have become a critical feature in the design of current scheduling systems. In other words, current scheduling systems should have the ability to incorporate new tasks that may arise after a scheduling plan already has been created. Therefore, this master thesis aims to incorporate into the ADSyS system an approach based on Artificial Intelligence techniques, to dynamically and in an efficient way perform this task. This way, the dynamic adaptation of a scheduling plan does not need human intervention. The system automatically deals with possible disturbances, in the best way possible, ensuring the usability of the system.

This approach consisted in implementing a hyper-heuristic based on an evolutionary algorithm, which searches for the near optimal combination of integration mechanisms to perform the integration process. Each integration mechanism consists in a different method to perform the integration of the new task, such as: EDD, ERD, FIFO, and GPF. A combination of these integration mechanisms identifies which machine in the scheduling problem uses

which mechanism. For example in a scheduling problem with 3 different machines the first machine may use the ERD integration mechanism, the second machine may use the GPF mechanism and the third machine may use the FIFO mechanism. Since the search space in which the evolutionary algorithm operates consists in a set of constructive low-level heuristics, and the algorithm selects the most suitable low-level heuristic at each stage, the implemented algorithm can be called a selective constructive hyper-heuristic. The evolutionary algorithm chosen was the genetic algorithm, since it has been successfully applied to the resolution of optimization problems (Pillay, 2014).

Furthermore, in this work a set of experiments were performed in order to evaluate the applicability of this proposed approach in the attempt of performing the integration process when compared with the approach already implemented into the ADSyS system (supervised learning approach). The major goal of this work was to validate if it could be an effective way to deal with dynamic events. Therefore, this new approach was evaluated against the supervised learning approach, and the results evaluated using performance metrics commonly used to analyze the performance of dynamic scheduling systems: utility, stability and robustness.

In general, from the obtained results of the performed experiments, it is possible to conclude that the proposed hyper-heuristic approach can be advantageous, at least with respect to the number of experiments performed, although no statistical significance evidence has been proved. The results obtained with proposed hyper-heuristic approach and with the supervised learning approach are very similar. Therefore, this approach can be evolved and refined for real world scheduling problems. Additionally, this approach can be more flexible, since each machine can use a different integration mechanism, instead of using the same integration mechanism in all the machines, as it happens in the supervised learning approach.

With this work, an important contribution has been proposed, allowing the ADSyS system to efficiently and effectively adapt and incorporate dynamic tasks into the current scheduling plan. Furthermore, this work was an excellent opportunity to acquire solid knowledge about dynamic scheduling systems, and different ambient intelligence techniques. Besides the achieved results, this unique and captivating area contains much more knowledge to be learned and many more challenges to explore in the future.

9.2 Open Issues and Future Work

Even though the experiments provide satisfactory results in relation to the selective constructive hyper-heuristic approach, there are some limitations and optimizations that should be done. This section identifies these limitations and proposed optimization approached for them.

The major issue found in the proposed selective constructive hyper-heuristic approach was the computational cost. As this hyper-heuristic is based on evolutionary computation, it iteratively searches for the near-optimal solution in a rather large space of candidate solutions.

In addition, as the size of the candidate solutions are dependent of the number of machines presented in the scheduling problem, the computational cost can be rather large, and the searching process can take a very long time.

As future work, it is proposed to perform a study about the parameters to be used in the genetic algorithm, in order to optimize its performance. In addition, the number of generation and population size can be dynamic, in other words, this parameters should adapt to the problem at hand to achieve better results.

References

Abraham, A. (2005). Evolutionary Computation. *Handbook of Measuring System Design*.

Alexander, F. J. (2013). Machine Learning. *Computing in Science & Engineering*, *15*(5), 9–11. doi:10.1109/MCSE.2013.107

Alpaydin, E. (2004). *Introduction to machine learning*. The MIT Press. Retrieved from http://books.google.com/books?hl=en&lr=&id=1k0_-WroiqEC&oi=fnd&pg=PR13&dq=Introduction+to+Machine+Learning&ots=paYDZQhEtL&sig=eA9Q7IgjPgHyFbyKJ-WxoMIFYeM

Ark, W., & Selker, T. (1999). A look at human interaction with pervasive computers. *IBM Systems Journal*, *38*, 504–507.

Ayodele, T. O. (2010). Machine Learning Overview. *New Advances in Machine Learning*, 9–19.

Baker, K. R., & Trietsch, D. (2009). *Principles of Sequencing and Scheduling*. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Ben-Hur, A., Ong, C., Sonnenburg, Ö., Schölkopf, B., & Rätsch, G. (2008). Tutorial: Support Vector Machines and Kernels for Computational Biology. *Computation Biology*.

Bick, M., & Kummer, T.-F. (2010). Ambient Intelligence. *Business & Information Systems Engineering*, *2*(5), 317–320. doi:10.1007/s12599-010-0117-5

Bock, C., Görlich, D., & Zuhlke, D. (2006). Using Domain-Specific Languages in the Design of HMIs : Experiences and Lessons Learned. *Model-Driven Development of Advanced User Interfaces*.

Boulesteix, A., Janitza, S., Kruppa, J., & Inke, K. (2012). Overview of Random Forest Methodology and Practical Guidance with Emphasis on Computational Biology and Overview of Random Forest Methodology and Practical Guidance with Emphasis on Computational Biology and Bioinformatics. *WIREs Data Mining & Knowledge Discovery*, (129).

Breiman, L. (2001). Random forests. *Machine Learning*, 1–33.

Burke, E., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, *64*(12), 1695–1724. doi:10.1057/jors.2013.71

Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., & Schulenburg, S. (2003). HYPER-HEURISTICS : AN EMERGING DIRECTION IN MODERN SEARCH TECHNOLOGY. In *Handbook of Metaheuristics*.

Burke, E. K., Hyde, M., & Kendall, G. (2010). A Genetic Programming Hyper-Heuristic Approach for Evolving 2-D Strip Packing Heuristics, *14*(6), 942–958.

BURKE, E. K., KENDALL, G., OCHOA, G., MISIR, M., OZCAN, E., & WOODWARD, J. (2010). Handbook of meta-heuristics, international series in operations research & management science. In *Classification of Hyper-heuristic Approaches* (pp. 449–468). New York: Springer.

Burke, E. K., Petrovic, S., Qu, R., Optimisation, A. S., & Campus, J. (2006). Case-based Heuristic Selection for Timetabling Problems. *Journal of Scheduling*, *9*(2), 115–132.

Burke, E. K., Silva, J. D. L., & Soubeiga, E. (2005). Multi-objective Hyper-Heuristic Apporaches for Space Allocation and Timetabling. *School of Computer Science and Information Technology*.

Calo. (n.d.). CALO Explanation Project Page. Retrieved April 14, 2014, from http://ksl.stanford.edu/projects/CALO/

Camps-valls, G., Member, S., Marsheva, T. V. B., & Zhou, D. (2007). Semi-Supervised Graph-Based Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, *45*(10), 3044–3054.

Chaudhary, A., Kolhe, S., & Kamal, R. (2012). Machine Learning Techniques for Mobile Intelligent Systems: A study. *2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN)*, 1–5. doi:10.1109/WOCN.2012.6335538

Commission, I. A. G. of the E. (2001). ISTAG scenarios for ambient intelligence in 2010.

Cook, D. J., Augusto, J. C., & Jakkula, V. R. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, *5*(4), 277–298. doi:10.1016/j.pmcj.2009.04.001

Cowling, P., Kendall, G., & Soubeiga, E. (2002). Hyperheuristics : A Tool for Rapid Prototyping in Scheduling and Optimisation. In *Applications of Evolutionary Computing* (pp. 1–10).

Cowling, P., Ouelhadj, D., & Petrovic, S. (2004). Dynamic scheduling of steel casting and milling using multi-agents. *Production Planning & Control*, *15*(2), 178–188. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/09537280410001662466

Darwin's Theory of- A Theory in Crisis Evolution. (n.d.). *http://www.darwins-theory-of-evolution.com/*.

Dittman, D. J., Khoshgoftaar, T. M., Wald, R., & Napolitano, A. (2013). Simplifying the Utilization of Machine Learning Techniques for Bioinformatics. *12th International Conference on Machine Learning and Applications*, 396–403. doi:10.1109/ICMLA.2013.155

Elomaa, T., & Malinen, T. (2003). On Lookahead Heuristics in Decision Tree. *Springer LNCS*, *2871*, 445–453.

Fathima, S., & Hundewale, N. (2012). Comparitive analysis of machine learning techniques for classification of Arbovirus. *IEEE-EMBS International Conference on Biomedical and Health Informatics*, *25*, 376–379.

Fogel, L. (1966). *Artificial Intelligence Through Simulated Evaluation*. *Artificial Intelligence Through Simulated Evaluation*.

Galitz, W. O. (2007). *The Essential Guide to User Interface Design* (3rd ed.). Wiley.

Garis, H. (1990). Genetic programming: Modular neural evolution for darwin machines. *International Joint Conference on Neural Networks*.

Gkoutioudi, K., & Karatza, H. D. (2012). A Simulation Study of Multi-criteria Scheduling in Grid Based on Genetic Algorithms. *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, 317–324. doi:10.1109/ISPA.2012.48

Hagras, H., Callaghan, V., Colley, M., & Clarke, G. (2004). Creating an ambient-intelligence environment using embedded agents. *IEEE Intelligent Systems*, *19*(6), 12–20.

Hansmann, U., Merk, L., Nicklous, M., & Stober, T. (2001). *Pervasive computing handbook*. Springer.

He, L., Liu, Y., Xie, H., & Zhang, Y. (2008). Job Shop Dynamic Scheduling Model Based on Multi-Agent. *Control and Decision Conference, …*, 829–833.

Hix, D., & Hartson, H. (1993). *Developing user interfaces: Ensuring Usability Through Product & Process* (1st ed.). Canada: Wiley.

Holland, J. (1975). Adaptation in Natural and Artificial Systems. *The University of Michigan Press: Ann Arbor*.

Hussmann, H., Meixner, G., & Zuehlke, D. (2011). *Model-Driven Development of Advanced User Interfaces*. Berlin: Springer.

Intelligence, A. (n.d.). IST Advisory Group Ambient Intelligence : from vision to reality For participation – in society & business, 1–31.

Jensen, M. (2001). Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures. *Applied Soft Computing*, *1*, 35–52.

Jensen, M. T. (2001). Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures. *Applied Soft Computing*, *1*(1), 35–52. doi:10.1016/S1568-4946(01)00005-9

Karray, F., Alemzadeh, M., Saleh, J. A., & Arab, M. N. (2008). Human-Computer Interaction: Overview on State of the Art. *International Journal on Smart Sensing and Intelligent Systems*, *1*(1), 137–159.

Kumar, V. (2002). Parallel Issues in Data Mining. *VECPAR*.

Madureira, A., Cunha, B., Pereira, J. P., Pereira, I., & Gomes, S. (2014). An Architecture for User Modeling on Intelligent and Adaptive Scheduling Systems. *Nature and Biologically Inspired Computing*.

Madureira, A., Gomes, S., Cunha, B., Pereira, J. P., Santos, J. M., & Pereira, I. (2014). Prototype of an Adaptive Decision Support System for Interactive Scheduling with MetaCognition and User Modeling Experience. *Nature and Biologically Inspired Computing*.

Madureira, A., Pereira, I., & Sousa, N. (2010). Collective Intelligence on Dynamic Manufacturing Scheduling Optimization. *EEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications*.

Madureira, A., Santos, J. M., Gomes, S., Cunha, B., Pereira, J. P., & Pereira, I. (2014). Manufacturing Rush Orders Rescheduling: a Supervised Learning Approach. *Sixth World Congress on Nature and Biologically Inspired Computing (NaBIC)*, 299–304.

Maeda, E., & Minami, Y. (2006). Steps toward ambient intelligence. *NIT Technical Review*, *4*(1).

Mallios, N., Papageorgiou, E., & Samarinas, M. (2011). Comparison of Machine Learning Techniques using the WEKA Environment for Prostate Cancer Therapy Plan. *IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 151–155. doi:10.1109/WETICE.2011.28

Mendelowitz, E., & Burke, J. (2005). Kolo and Nebesko: A Distributed Media Control Framework for the Arts. *First International Conference on Distributed Frameworks for Multimedia Applications*, 113–120. doi:10.1109/DFMA.2005.33

Michalewicz, Z. (1996). Heuristic Methods for Evolutionary Computation Techniques. *Journal of Heuristics*, *1*(2), 177–206.

Mitchell, M., & Taylor, C. E. (1999). Evolutionary Computation: An Overview. *Annual Revies Ecol. Systems*, *30*, 593–616.

Mitchell, T. (1997). Machine Learning. *McGraw-Hill*.

Mitchell, T. (2006). he Discipline of Machine Learning. *Technical Report CMU-ML-06-108*.

Mozer, M. . (2004). Lessons from an adaptive home. In *Smart Environments: Technology, Protocols, and Applications*. Wiley.

Nielsen, J. (1993). *Usability engineering*. California: Morgan Kaufmann.

Ouelhadj, D., & Petrovic, S. (2007). Suvey of Dynamic Scheduling in Manufacturing Systems (Vol. 44, pp. 1–27). Wiley.

Ozcan, E., Misir, M., & Kheiri, A. (2013). Group Decision Making Hyper-heuristics for Function Optimisation. *Computational Intelligence (UKCI)*, 327–333.

Panait, L., & Luke, S. (2005). Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems*, *11*(3), 387–434. doi:10.1007/s10458-005-2631-2

Phillips. (2007). Other perspectives on ambient intelligence.

Pillay, N. (2014). Evolutionary Algorithm Hyper-Heuristics. *Nature and Biologically Inspired Computing*.

Pinedo, M. (2005). *Planning and scheduling in manufacturing and services*. (P. W. Glynn & S. M. Robinson, Eds.). Springer.

Priore, P., Fuente, D. D. La, Gomez, A., & Puente, J. (2001). A review of machine learning in dynamic scheduling of flexible manufacturing systems. *AI EDAM*, 1–41.

Rechenberg, I. (1973). *Evolution Strategie*.

Ross, P. (2005). *Hyper-heuristics. Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. (E. K. Burke & G. Kendall, Eds.). Springer.

Sabuncuoglu, I., & Goren, S. (2009). Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. *Computer Integrated Manufacturing*, 1–41.

Sadri, F. (2011). Ambient Intelligence: A Survey. *ACM Computing Surveys*, *43*(4), 1–66. doi:10.1145/1978802.1978815

Srinivas, M. (1994). Genetic algorithms: a survey. *EEE Computer Society*, *27*(6).

Vasilakos, A., & Pedrycz, W. (2006). *Ambient Intelligence, Wireless Networking, and Ubiquitous Computing*. Artech House Publishers.

Vieira, G., Herrmann, J., & Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of Scheduling*, *6*, 39–62. Retrieved from http://link.springer.com/article/10.1023/A:1022235519958

Vijaya, M. ., Jamuna, K. ., & Karpagavalli, S. (2009). Password Strength Prediction Using Supervised Machine Learning Techniques. *International Conference on Advances in Computing, Control, and Telecommunication Technologies*, 401–405. doi:10.1109/ACT.2009.105

Weiser, M. (1991a). The computer for the twenty-first century. *Scientific American*, *165*, 94–104.

Weiser, M. (1991b). The computer of the twenty-first century. *Scientific American*, *265*, 66–75.

Weyrich, C. (2000). Information Society Technologies Advisory Group Orientations for Workprogramme 2000 and beyond. Retrieved April 25, 2014, from ftp://ftp.cordis.europa.eu/pub/ist/docs/istag-99-final.pdf

Xue, M., & Zhu, C. (2009). A Study and Application on Machine Learning of Artificial Intellligence. *International Joint Conference on Artificial Intelligence*, 272–274. doi:10.1109/JCAI.2009.55

Yingzi, W., Xinli, J., & Pingbo, H. (2009). Pattern driven dynamic scheduling approach using reinforcement learning *. *International Conference on Automation and Logistics Shenyang*, (2002), 514–519.

Zhang, H., Lenaghan, S. C., Connolly, M. H., & Parker, L. E. (2013). Zebrafish Larva Locomotor Activity Analysis Using Machine Learning Techniques. *12th International Conference on Machine Learning and Applications*, 161–166. doi:10.1109/ICMLA.2013.35