U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Evaluating agile scheduling methods for a job shop problem

Ricardo de Sá Caetano Ferreira da Cunha

Mestrado Integrado em Engenharia e Gestão Industrial
Master's Dissertation

Supervisor: Prof. Pedro Sanches Amorim
Co-supervisor: Prof. Luís Gonçalo Reis Figueira

2017-06-25

"It is not the strongest of the species that survives, nor the most intelligent, but the one most responsive to change"

Charles Darwin, 1809

Resumo

As exigências de um mercado cada vez mais global e a evolução tecnológica forçada pela visão da Indústria 4.0, impõem a necessidade de novas ferramentas de tomada de decisão, capazes de agir num espaço de tempo quase imediato. As técnicas tradicionais de controlo da tomada de decisão são maioritariamente baseadas em estruturas centralizadas, o que representa inúmeras desvantagens quando o sistema tem de tomar decisões ágeis e quase imediatas, de forma a adaptar-se aos vários tipos de eventos imprevistos. Nesse sentido, a descentralização da tomada de decisão pode ser uma solução interessante para ultrapassar esses problemas.

Esta dissertação trata o problema de *job-shop scheduling*, em que várias técnicas ágeis e descentralizadas são utilizadas e avaliadas, de forma a perceber qual é a perda de qualidade da solução quando se opta por este tipo de métodos em detrimento de algoritmos centralizados mais complexos. Para isso, foi desenvolvido um modelo de simulação adaptável a qualquer ambiente *job-shop*, que foi validado com a aplicação dos problemas estudados por Lawrence and Sewell (1997) (L&S). No sentido de estender a investigação feita por L&S, os métodos que são utilizados no seu artigo, juntamente com métodos mais avançados orientados para a otimização do tempo de *setup*, são implementados em instâncias de um problema real, que são inevitavelmente mais complexas e envolvem *setups* dependentes da sequência. De forma a estudar a influência desta última característica, as mesmas técnicas de *scheduling* são aplicadas aos problemas estudados por L&S, mas em que são gerados quatro níveis de tempos de *setups* para cada problema. Finalmente, utilizando um modelo de programação genética, é desenvolvido um método ágil e descentralizado, capaz de se adaptar às características do problema em que é aplicado.

Os resultados das experiências simuladas nas instâncias reais permitiram concluir que os métodos orientados para a otimização do tempo de *setup* têm resultados significativamente melhores do que as técnicas estudadas por L&S. Relativamente ao estudo do impacto to tempo de *setup*, foi demonstrado que o aumento do nível do tempo de *setup* melhora a performance relativa das regras orientadas ao tempo de *setup*. No entanto, mesmo para níveis elevados de *setup*, as técnicas utilizadas por L&S obtêm maioritariamente os melhores resultados nos problemas quando aplicadas às instâncias de referência usadas por estes autores, mas com *setups* gerados. Finalmente, apesar do esforço computacional envolvido, a técnica desenvolvida utilizando programação genética demonstra um bom comportamento, não só face à presença de *setups* dependentes da sequência elevados, mas também revela uma grande robustez à variação da incerteza dos tempos de processamento.

Abstract

The new requirements of the global market and the new technological developments in light of the vision of Industry 4.0 demand new decision-making tools capable of acting on a real-time basis. Traditional production process modelling techniques rely heavily on central decision-making structures, which present numerous disadvantages when they have to deal with real systems, whose agility and responsiveness are fundamental to manage all kind of disturbances. Decentralization of decision-making may be an interesting solution to overcome these issues.

This dissertation is focused on a job-shop scheduling problem and involves the implementation and evaluation of the performance of agile and decentralized methods, analyzing the loss of quality in comparison with centralized and more complex algorithms. To achieve this purpose, a simulation model to reproduce any job-shop environment is developed. This model is validated with the application to the instances studied by Lawrence and Sewell (1997) (L&S). In order to extend the investigation performed by L&S, the methods proposed in their research together with more advanced setup-oriented methods are implemented in real-world case problems, which are much more complex and involve sequence-dependent setup times. Aiming at investigating the influence of sequence-dependent setup times, the same scheduling methods are applied considering four levels of setups are generated to the benchmark instances studied by L&S. Finally, an evolved agile and decentralized method capable of adapting to the characteristics of the problem is developed using a genetic programming model.

From the experiments performed with the real-world instances, it is concluded that setup-oriented methods achieve significantly better performance than the methods studied by L&S. Regarding the investigation of the impact of the setup times, it is shown that as setup time increases, setup oriented methods achieve better performance. However, even for high levels of setups these methods do not lead to better results than the others applied in the benchmark instances of L&S. Finally and despite the heavy computational effort, the evolved method using the genetic programming technique shows not only a good behavior in presence of large sequence-dependent setup times, but also a good robustness against processing times' uncertainty.

Acknowledgements

I would like to thank my supervisor, Prof. Pedro Amorim, for the opportunity to work with him and for the crucial definition of the guidelines that supported the development of this master project. His competences and enthusiastic commitment have served me as inspiration throughout this thesis.

I would also like to thank my co-supervisor, Prof. Gonçalo Figueira, who closely followed the progress of this project. His expertise and knowledge were fundamental to help me not only overcome the obstacles along the way, but also to open up my horizons, introducing me a wide range of new concepts and techniques. But, his support went beyond the technical assistance, and he has become a good friend along this time.

I am also grateful to my colleagues from CEGI, who not only helped me in the different phases of this work with some technical advices, but also were an excellent company during the lunch times, contributing to enrich me as person.

I would also like to thank to my friends, whose support was again essential to accomplish one more step of my evolution.

Finally, I want to thank my family, with a special reference to my parents. Mãe and Pai you are my examples and the support that helped me getting where I am today.

Table of contents

1 Introduction 1
1.1 Motivation 1
1.2 Objectives 2
1.3 Methodology 2
2 Theoretical framing 5
2.1 History of manufacturing 5
2.2 Industry 4.0 5
2.3 Centralization vs decentralization 6
2.3.1 Advantages and limitations 8
2.3.2 Best fit scenarios and applications 8
2.4 Scheduling 9
2.4.1 Scheduling problem description 9
2.4.2 Scheduling problem classification 11
2.5 Scheduling methods 14
2.5.1 Optimal approaches 14
2.5.2 Non-optimal approaches 16
2.6 Simulation 21
2.6.1 Methods of simulation 21
2.6.2 Simulation for scheduling problems 22
2.6.3 Simulation-optimization 22
2.7 Relevant techniques for this research 23
3 Description of case studies 25
3.1 Lawrence and Sewell's research 25
3.1.1 Results and conclusions from the paper 26
3.2 Real-world case 27
3.3 Description of the problem under study 28
3.3.1 Problem definition 28
3.3.2 Characterization of the scheduling problem 28
4 Model and experiments' design 31
4.1 Simulation model 31
4.1.1 Selection of the simulation method 31
4.1.2 Model components 31
4.2 Experiments' design 35
4.2.1 Introduction of uncertainty 35
4.2.2 Static vs dynamic experiments 36
4.2.3 Lower Bounds 36
4.3 Implemented scheduling methods 37
4.3.1 Scheduling methods from Lawrence and Sewell 37
4.3.2 Advanced setup-oriented scheduling methods 38
4.4 Sequence-dependent setup times' generation 41
4.5 Validation of the simulation model 42
5 Analysis of results 45
5.1 First experiments' test 45
5.1.1 Results 45
5.1.2 Discussion of the results 46
5.2 Second experiments' test 49
5.2.1 Results 49
5.2.2 Discussion of results 50
6 Evolved Methods 55
6.1 Method design 55

6.2 Rule evolved 56
6.3 Tests and discussion of results 57
7 Conclusions and future work 59
References 61
Appendix A: Mean Performance of the methods applied to the benchmark intances for both static and dynamic approaches 67
Appendix B: Mean Performance of the methods applied to the real-world instance for both static and dynamic approaches 68
Appendix C: Mean makespan and production capacity for the benchmark instances with setup level generated of 1 69
Appendix D: Mean makespan and production capacity for the benchmark instances with setup level generated of 0,75 70
Appendix E: Mean makespan and production capacity for the benchmark instances with setup level generated of 0,5 71
Appendix F: Mean makespan and production capacity for the benchmark instances with setup level generated of 0,25 72
Appendix G: Mean makespan performance of the method evolved using genetic programming 73

List of Figures

Figure 1: Control architectures' scheme (Dilts, Boyd and Whorms, 1991) 7
Figure 2: Setting of Predictive Reactive Scheduling 13
Figure 3: Branch and Bound tree (Pinedo, 2008) 15
Figure 4: Disjunctive graph representation (Barzegar, Motameni and Bozorgi, 2012) 17
Figure 5: Crossover process of a mathematical expression (Geiger, Uzsoy and Aytuğ, 2006) 20
Figure 6: Tree Hierarchical representation of a computer program generated by a GP (Ho and Tay, 2005) 20
Figure 7: Agent-Based Modelling Scheme (Anylogic Web Page, 2017) 22
Figure 8: Comparison of the mean makespan performance of fixed (left) and dynamic (right) sequence scheduling methods (based on the research o L&S) 27
Figure 9: State chart of the order agent 32
Figure 10: State chart of the machine agent 33
Figure 11: Environment of main agent during the simulation 34
Figure 12: Diagram of the functional mechanism of the interaction between agents 35
Figure 13: Encoding Process of the genetic algorithm 38
Figure 14: Flowchart of the genetic algorithm implemented algorithm 39
Figure 15: Pseudocode of the Constructive Heuristic scheduling method 40
Figure 16: Dispersion of the processing time of the set of 53 instances 42
Figure 17: Comparison of the mean makespan performance fixed (left) and dynamic (right) sequence scheduling methods in the research of L&S (top) and in obtained results (bottom). 43
Figure 18: Difference of mean performance of static and dynamic methods for instances of L&S(left) and real-world instances (right) 47
Figure 19: Mean performance of static and dynamic methods for real-world instances 47
Figure 20: Loss of quality of scheduling methods in real-world instances and in the research of L&S 48
Figure 21: Mean performance of each scheduling method for each setup level 50
Figure 22: Relative ranking of scheduling method for each setup level 51
Figure 23: Loss of quality relatively to the most optimist lower bound at each setup level 52
Figure 24: Evolved rule using Genetic Programming 56
Figure 25: Quality of the solutions evaluated in the Genetic Programming model 57
Figure 26: Mean performance of the scheduling method for the benchmark instances with setup level of 0,25(left), 0,5(center) and for the instances of the real-world case (right) 57

List of Tables

Table 1: Size of the instances studied in the paper 25

Table 2: Mean performance of scheduling methods applied in the paper of L&S 26

Table 3: Size of the real-world case's instances 28

Table 4: Dispatching Rules implemented in the simulation model 38

Table 5: Results obtained for the various scheduling methods and their differences from the paper of L&S 43

Table 6: Results obtained for each scheduling method in the first set of experiences 46

Table 7: Mean performance of no setup oriented methods for real-word instances and in L&S research 48

Table 8: Mean scheduling methods' performance of all levels of uncertainty methods for each setup level 49

Table 9: Percentage of working time and relative ranking of each method for the four setup levels investigated 52

1 Introduction

1.1 Motivation

In the last decades, the world has moved towards a global economy, building a market dynamic that fosters competition and thus present new requirements to producers. Nowadays, companies must cope with customers demanding high quality products at lower costs, highly customized and with short life cycles (Leitão, 2009). Hence, processes in supply networks have to be constantly re-configured to adapt to the new changing conditions. In fact, productivity, flexibility and agility to react to market demands are more than ever critical to manufacturing.

In 2000s, the advantages of outsourcing and offshoring, moving low-skill manufacturing to low-cost countries began to shrink, which forced companies to invest in automation and robotics technologies with potential to reduce production costs and boost productivity (Wee *et al.*, 2015). There is no doubt that the advances in technologies with high flexibility, such as robotics, will play an important role in future manufacturing, but its impact in the overall decision making is still an open question.

This conjuncture of forces has emerged the concept of Industry 4.0 which conceives the idea of Cyber-Physical Systems (CPS), interlinking both digital and physical world, communicating through the Internet of Things and Services (Waschneck *et al.*, 2016). In this environment, robots are able to collaborate with humans in a shared workspace in the shop-floor, and receive manufacturing orders in a completely flexible manufacturing scenario.

To turn this vision into reality, it is necessary to improve decision making tools to develop systems able to provide good and robust decisions in reduced time frames, when processing large data sets of information. This new paradigm represents a challenge for the traditional production process modelling techniques, built upon centralized structures that present good static optimized production solutions, but a weak capacity to response to disturbances.

For real-world manufacturing, finding the right sequences and allocation of limited resources to operational jobs over time is a difficult task, since generally the complexity of the problem increases exponentially. The problem is even more complicated in a dynamic environment, subject to a high level of uncertainty where several unexpected events can happen and the new manufacturing paradigm amplifies even more this issue. Thus, novel optimization methods, simulation or simulation based optimization tools are required to, not only generate robust production schedules, but also to undertake real-time rescheduling to cope with the production environment uncertainty.

1.2 Objectives

In the past few years several dynamic scheduling methods have been presented to deal with the occurrence of real-time events (Ouelhadj and Petrovic, 2009). But the vision of Industry 4.0, conceptualizing smart factories with CPS interconnected in an Internet of Things, increases the complexity of the production. Furthermore, uncertainties due to the volatile demand and shop floor disturbances are much more likely to happen due to today's market conjecture. In these circumstances, corrections of the planned schedule are difficult to manage, and optimizing the production flow at a central level may become impossible at some point (Wee *et al.*, 2015).

In this context, decentralized scheduling methods are an interesting solution, since they assure agile and flexible responses in a short timeframe. However, they can disregard the quality of the solution, as there is a high likelihood that most of the times the scheduling optimization is only done at a local level, rather than global.

There are numerous operational issues that can have impact on scheduling, impeding the improvement of capacity management and hindering greater competitiveness. One of the most common factors with major implications in scheduling is sequence-dependent setup times, which significantly increases the complexity of the problem. Usually, these problems require complex central heuristics to generate effective schedules.

Against this background, the fundamental research questions of the present master thesis are the following:

1. Which cases motivate a decentralized decision-making structure?
2. What is the quality loss of decentralized scheduling methods when compared to centralized solutions?
3. What is the advantage of performance of dynamic over the static scheduling methods?
4. What are the implications of sequence-dependent setup times on the quality of decentralized and methods?
5. Are sequence dependent setup times a critical factor when optimizing for decentralize methods?
6. Which scheduling methods can better cope with both sequence-dependent setup times and uncertainty variation?

1.3 Methodology

The approach to address the above presented research questions is divided into two fundamental parts.

Part 1: Theoretical foundation:

In the context of the actual conjuncture and in the future prospect of manufacturing, the theoretical foundation aims at explaining what motivates the choice for a centralized or decentralized decision-making structure. A qualitative analysis to assess the advantages and limitations of both centralized and decentralized control approach is performed. Focusing on a decentralized decision, an investigation is made of the scenarios that can benefit the most from this control structure. Additionally, this part of the thesis aims at providing a review of the scheduling problem's classifications and the most common solving methods. A review of the simulation techniques and their applications to scheduling is also presented.

Part 2: Case-study and empirical research:

This research extends the work of Lawrence and Sewell (1997) (L&S) which provides a good investigation about the performance of static and dynamic methods to job-shop problems. In the present research, those methods are applied to a real-world case of a Portuguese company in order to validate the conclusions presented in the above-mentioned paper.

The instances of this real-world complex case have sequence-dependent setup time, which constitutes a major difference from problems studied in the work of L&S. Thereby, more advanced methods are implemented and their performance is evaluated, establishing a comparison with those studied in L&S's paper.

The methodical approach for the research is conducted using an agent-based simulation model to predict the performance of the scheduling techniques in a simulated scenario. Two sets of experiments are performed aiming at different objectives. In the first, dispatching rules used in Lawrence and Sewell (1997) as well as more advanced setup-oriented techniques are evaluated under several processing times' uncertainty levels for the two sets of problems (L&S and the real-world case). In these tests the goal is to understand whether the conclusions from L&S's work about both the static and dynamic methods and the performance of the scheduling methods remain valid for larger and more complex problems with sequence-dependent setup times.

Subsequently, the objective of the second set of tests is to understand the impact of sequence-dependent setup times on the applied agile scheduling methods. Thereby, four setup levels are established. Four each level setup times' matrices are generated in correspondence with each problem studied in the research of L&S. In these experiments both techniques applied by L&S and the more advanced setup-oriented methods are implemented and evaluated. The results of these tests are analysed and compared with those of the real-world case to examine both the implication of the setup level on the performance of the scheduling method and the influence of sequence-dependent setup times as a critical factor in decentralized scheduling.

Finally, an adaptive method, which looks to several problems characteristics to be properly adjust, is evolved through genetic programming. The final objective is to formulate simple and agile rule that adapt to the specific features of the problem.

2 Theoretical framing

2.1 History of manufacturing

The manufacturing industry we know today has been shaped by some radical transformations along the times.

Before the 18th century, craft production was dominant, with skilled workers using general purpose tools to produce exactly what customers wanted. Everything was done manually and the quality of the products was heavily dependent on the ability of the employees. At that time, the creation of the first steam engines and the intelligent use of hydropower, enabled to introduce machinery in production to support craftsman work, improving their productivity in a way that started the first industrial revolution.

The second period of radical transformations occurred in the beginning of the 20th century when the use of electrical power replaced coal and steam power, which allowed engineers not only to make important technological developments, but also to rethink the division of labour. This established a paradigm shift in manufacturing, with factories mass producing in large assembly lines. This era was epitomized by Henry Ford through the creation of a line to build one single car model composed by identical interchangeable parts.

However, mass production requires stability and control of the production variables, markets and the labour force. And in 1970s, globalization made these parameters become less stable with the globalization. The global competition started to be fiercer and fiercer, which altered the homogeneity of the market, since consumers had increased their power. To obtain economies of scale and flexible production, meeting the volatile demand of the markets, producers had to cope with more unpredictable and demanding customers. Mass production of durable consumer goods had to be changed, by increasing the quality of products and at the same time diversifying them with multiple models and other optional features.

The third revolution emerged in these years, when digital technology such as electronics and information technology began to expand rapidly into industry. The advent of microprocessors brought automation into plants on a large scale, and production become increasingly based on computer system controls. Through information and communication technology, new flexible automation solutions were developed and manufacturing production became more flexible, automated and responsive to markets.

2.2 Industry 4.0

In the 1990s, outsourcing and offshoring allowed companies to enjoy of greater profitability, by moving low-skilled manufacturing to low-cost countries. But in the 2000s the wages of those countries rose and freight costs increased, which began to shrink the margins of those strategies (Wee *et al.*, 2015). So, industry players are now forced to find other competitive advantages.

In 2011, the vision of the fourth Industrial revolution emerged as an approach to strengthening the competitiveness of manufacturing industries in different areas (Hermann, Pentek and Otto, 2016). In contrast with the other radical transformations that happened along the history, this new revolution is expected to become a reality in the next 10 to 20 years. In fact, Industry 4.0 is still a vision that describes a highly flexible control of production in which Cyber-Physical Systems (CPS) are connected in an Internet of Things to communicate with each other and more important to control each other in real time (Hermann, Pentek and Otto, 2016).

Therefore, there is no need of a central control system to manage production. Instead, decentralized and autocratic operating systems run locally and independently, monitoring the physical processes of the factory. Those systems are able not just to self-optimize, but also, by establishing communications and cooperating with other agents, to optimize the production as a whole.

The optimization process is possible due to the virtual replication of the physical world through sensor data, which capture real time information to measure the performance of the actual processes. Then the necessary adjustments can be made to achieve the ideal processes calculated by the system. Also, the spatial decoupling of physical assets and their monitoring allows for more agility and flexibility to adapt and respond to disturbances in production ((Wee *et al.*, 2015). In fact, the integration of CPS and the Internet of Things allows to establish smart factories, capable of better managing complexity, and thus produce more efficiently.

On the other hand, the integration of CPS, making decentralized decisions, form complex networks, which increases the complexity of production. Thus, these systems need to aggregate information comprehensively to support and collaborate with human operators. This interaction is particularly important, since the role of human in production is altered in the vision of Industry 4.0. In fact, humans are not merely operators, but they must be a strategic decision-makers, working together with machines that conduct the exhausting and unsafe tasks (Hermann, Pentek and Otto, 2016).

Nevertheless, as any major shift, there are challenges inherent to these transformations. These issues according to (Wee *et al.*, 2015) are mainly related with the lack of process and control know-how of the employees; data security and safeguarding systems due to the integration and information and data sharing; and the need of a uniform standard for data transfer.

But the benefits of an Industry 4.0 model can outweigh the concerns for many production facilities. In truth, for producers, it may mean highly flexible mass production with the agility needed to easily adapt to market changes. Also, supply chains can be more readily controlled when there is data at every level of the manufacturing process. In the customers' perspective, they may benefit from the access to tailored made products at relatively affordable prices.

2.3 Centralization vs decentralization

In the last decades, globalization has changed the world fundamentally, shaping a global economy which transcend national and cultural borders. Nowadays, markets demand for products with higher quality at lower costs, highly customized and with shorter life cycles, which impose new requirements to manufacturing (Leitão, 2009). Companies are now forced to compete not only at the price and quality level, but they should also be able to be responsive and flexible to comply in the minimum time frame to customers' requests.

On the other hand, the need to create competitive advantages, have been pushing companies to embrace Industry 4.0, making use of the increasing digitalization and

networking of men and products with each other (Sauter, Bode and Kittelberger, 2015). In order to cope with the new demand requirements and with the increasing complexity of production systems, companies should rethink their organizational structure. (Brettel, Friederichsen and Keller, 2014).

In fact, traditional centralised control approaches, where all manufacturing activities and resource are managed by a single decision maker, are no longer the most convenient to cope with new production requirements. The fact that these control systems rely heavily on central entity presents numerous disadvantages when they have to deal with real systems, whose agility and responsiveness are fundamental to comply with demand requirements and manage all kind of disturbances. In order to tackle these new challenges, the integration on new technologies and control methods become necessary. (Hulsmann and Windt, 2007). This is why Industry 4.0 aims at shifting the paradigm from a centralized control of non-intelligent items towards a decentralized control of intelligent items in a distributed structure.

A decentralized control approach delegates the decision-making power to the organizational units on the shop floor, which make it easier to handle complexity, since decisions are taken at local level (Hulsmann and Windt, 2007). They reduce the number of necessary arithmetic operations and thus they require fewer computational efforts and are more time saving. On the other hand, a decentralized decision-making process requires the availability of relevant information for the system elements (Hulsmann and Windt, 2007). These capabilities seem particularly relevant to capture the full Industry 4.0 potential, since they allow for more agility and flexibility on production processes (Wee *et al.*, 2015) . The expected benefit of decentralized structures is that in case of increasing system complexity in combination with many disorders, autonomous local decisions are capable to improve the performance of the system as a whole (Hulsmann and Windt, 2007).

In the past years, manufacturing control structures have attracted the attention of researchers and several control architectures have been proposed. Dilts, Boyd, & Whorms (1991) consider four basic types of control architectures: centralized, hierarchical, modified hierarchical and heterarchical. In the centralized architecture, a single entity concentrates the decision-making power and is responsible for all planning functions. The hierarchical structure distributes decision power among the levels of the hierarchy, in an attempt to introduce a better robustness to the system. The modified hierarchical structure shares many similarities with the hierarchical one, but differs on the degree of autonomy of the subordinates, which enabled to interact at the same hierarchical level, increasing the expansibility of the system. Finally, the heterarchical architecture is composed by local entities which communicate with each other without a central authority. Figure 1 displays a graphical representation of each architecture's scheme.

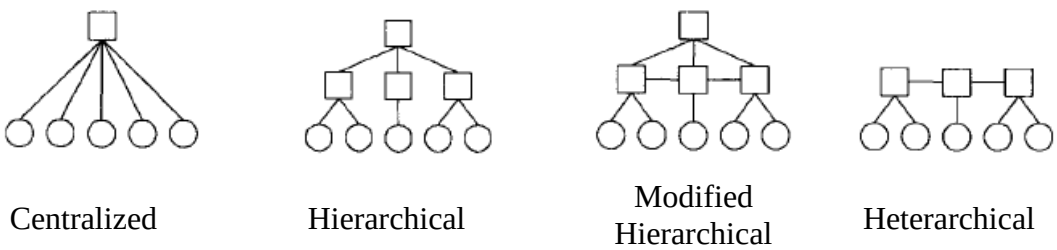

Figure 1: Control architectures' scheme (Dilts, Boyd and Whorms, 1991)

In recent years a new approach of decentralized control, entitled autonomous control, has been proposed to handle the increasing complexity and dynamics of the systems (Braun, 2014). This approach is characterized by the ability of autonomous and interacting entities to process information, to render and execute decisions in a non-deterministic system (Hulsmann

and Windt, 2007). Autonomy in decision-making is enabled by the alignment of the system elements in a heterarchical organizational structure. In fact, each entity is characterized by target-oriented-behaviour, and their objective can be dynamically updated during the production process. This approach seems particularly interesting to align manufacturing control and the vision of Industry 4.0.

2.3.1 Advantages and limitations

Effectively, both centralized and decentralized control concepts of decision-making have advantages and limitations, which should be assessed to understand the best-case application for each approach.

Advantages of centralized control are concerned to the easiness to access to global information, which allows not only to perceive the overall system status, but the most important to make optimization a reachable prospect. On the other hand, a centralized control reduces the speed of response, especially when the system gets larger. A second disadvantage is the reliance on a central unit, which means that if that unit fails, the whole system is affected. Finally, modifications on a central system tend to be hard to be implemented, since they can affect the dynamics of whole complex system.

Decentralized control structures evolved as a response to the disadvantages of centralization. Therefore, decentralized control enables to achieve a full local autonomy, which reduces complexity, increases flexibility and improves the fault-tolerance of the system. Also, local decisions involve less complexity, which speeds up the system response. However, there is a high likelihood that local entities have a greedy behaviour and, thus only optimize the system at a local level.

2.3.2 Best fit scenarios and applications

As stated in the previous section, both central a distributed decisions structures present advantages and limitations, which does not permit to find a common agreement about one mandatory control structure that best fits to every case scenarios.

However, it is possible to identify the critical factors that drive the decision about the degree of decentralization of the control structure. It is important to denote that the identified elements do not act alone, but it is the combination of their effects that should orientate the level centralization. First, the uncertainty on the shop floor level may require a different approach to re-configure the system to the constant alterations. This means that in environments subjected to production disturbances, such as machine failures or cancellation and arrival of rush orders, and high levels of variability, decisions must require a decentralize decision making structure that provides a quick reaction to the new conditions. A second critical factor is the time frame to find a new solution. As stated in the section above, centralized control tends to reduce the speed of response, particularly when the system complexity increases, since they decide based on the global information of the system. In production environments that demand real-time decisions, it is difficult to analyse centrally the data. Another critical factor that should be taken into account is the amount of information needed to be handled in the decision process. In fact, in systems where big amounts of data are generated and need to be analysed, central decision-making will become a mission impossible at some point.

To wrap up, it is the increasing need for flexibility and agility that drives the decentralization of intelligence (Wee *et al.*, 2015). Thus, in high complex manufacturing systems, where large amounts of data are generated, decisions should if possible to be taken locally. One industry that constitutes a good example of those manufacturing system is the semiconductor industry. Effectively, this type of industry is a prime example of a highly

complex system, in which the non-synchronization of supply and demand represents a critical challenge (Aelker, Bauernhansl and Ehm, 2013). Furthermore, large amounts of data are generated along the production which cannot be centrally analysed in real-time (Waschneck *et al.*, 2016). For these reasons, semiconductor is the perfect example of an industry where decisions should if possible be taken locally.

2.4 Scheduling

Scheduling is defined as a decision-making process used on a regular basis in manufacturing to allocate resources to tasks on a given period of time with one or more objectives (Pinedo, 2008). Consequently, it plays a major role in most production systems, since it can have a crucial impact on the productivity of the processes.

The scheduling problem in manufacturing started to be explored in the beginning of the 20th century with the work of Henry Gantt and other pioneers, but only in the 50s we find the first publications on this area, showing the results of W.E.Smith, S.M.Johnson and J.R.Jackson (Pinedo, 2008). Since then, a lot of authors have focused their attention on scheduling developing methods for both deterministic and stochastic scheduling problems.

The new shape of global market demand imposes new challenges to production. In fact, in a context of quick change in terms of demand variability, breakdowns, maintenances stoppages and production capacity availability, innovative scheduling solutions for efficient use in manufacture environment, are required. It's is not enough to create static optimal schedules with a perfect behaviour under a couple of conditions.

In today's conjuncture, schedules must present good solutions even subjected to the uncertainty of the production and other external issues. Solutions exploring real-time responses to every type of events with low computation burden are needed to keep coherency and optimize the decision-making.

In the next sub-sections, the descriptive notation used in scheduling and the classification of different types of problems is presented and explained.

2.4.1 Scheduling problem description

The notation proposed by Graham *et al.* (1979) describes scheduling problem by a triplet $\alpha|\beta|\gamma$. Field α defines the machine environment and contains just one entry. Field β details the characteristics and constraints of the process and can include none or multiple entries. Finally, field γ concerns the optimization criteria to be considered and can include one or more objectives.

According to Pinedo (2008) the possible machine environments to describe the α field are:

Single machine (1): In the case of a single machine all tasks are processed, one at a time, by a single resource;

Identical machines in parallel (Pm): In this environment, there are m identical machines and the job j requires a single operation in any machine that belongs to a given subset M_j;

Machines in parallel with different speeds (Qm): This environment is identical to the previous one, but the machines have different speeds, but they are independent of the jobs being processed;

Unrelated machines in parallel (Rm): This environment is a further generalization of the previous one. The difference is that different machines can process at different speeds the job j which creates a dependency between machines and jobs;

Flow shop (Fm): Jobs have m operations that must be process on m machines following the same route;

Flexible Flow Shop (FFc): this case is a generalization of the flow shop, but instead of m machines in series there are c stages in series. At each stage there are a number of identical machines in parallel to process the job. Jobs may skip some of the stages, but the all follow the same order;

Job Shop (J): In this case there are m different machines and each job has its own predetermined route to follow. Some machines may be missed;

Flexible Job Shop (FJc): This case is a generalization of a job shop problem and parallel machine environment. Instead of m machines in series, there are c work centres with a number of identical machines in parallel. Each job has its own route to follow through the shop and requires processing at each work centre on only on machine;

Flexible Job Shop (MRFJc): This case is a generalization of the job shop problem in which each job may have more than one route with a number of operations associated that are not necessarily equal (Golmakani and Birjandi, 2013);

Open Shop (O): Jobs must be processed once on each of the machines without a predetermined route.

Field β characterizes the processing restrictions and constraints between jobs and tasks and may include multiple entries. Some of the most common are exposed.

Release dates (rj): A job can only start after a given time t;

Preemptions (prmp): A job can be interrupted if there is a higher priority job to be processed;

Precedence contraints (prec): A given job J_k requires another job J_i to be completed before it is allowed to start its processing;

Sequence dependent setup times (sjk): The setup time of job J_k depends on the job J_i being processed on the machine. In these cases, setup times cannot be considered part of the processing times;

Batch processing (batch(b)): A machine may be able to process a number of jobs simultaneously and the setup time takes place before the production of each batch start. In this type of problems, jobs are divided into families and batches are created with jobs of the same family;

Breakdown (brkdwn): Machine breakdowns imply that a machine may no be continuously available;

Recirculation (rcrc): It occurs in a job shop or flexible job shop when a job may visit a machine or work centre more than once;

The last parameter to define the scheduling problem is the optimization criteria, γ. Some of the most common objectives are presented.

Makespan (C_{max}): Makespan is defined as the completion time of the last task processed. A minimum makespan usually implies a good utilization of the machines;

Lateness (L): Lateness is defined as the difference between the completion time of a job C_j and its due date d_j, $L_j = C_j - d_j$. It is used to minimize the worst divergence from the due dates;

Tardiness (T): Tardiness is defined as $T = \max(C_j - d_j, 0)$. If there is any late job the value of tardiness is the same as lateness. However, it should be highlighted that these indicators are very different: tardiness never assume negative values while lateness does. It is usually used to minimize the total tardiness of the jobs;

Earliness (E): Earliness, define as $E = \max(d_j - C_j, 0)$, is the opposite of tardiness. Usually this metric is used to minimize the inventories of finished products;

Production Capacity: Production capacity measures the units or time that machines are actually processing in a specific time horizon;

Weighted Criteria: In most of the real-world problems, it is important to take into account more than one metric, since there is not one single objective in production. Through a weighted criterion is possible to establish the priorities of the metrics considered and define one single objective function.

2.4.2 Scheduling problem classification

2.4.2.1 Deterministic vs Stochastic

The manufacturing scheduling becomes a complex combinatorial problem, more specifically a non-polynomial (NP) problems, for larger scheduling problems. For a generic problem with n jobs for m machines, the number of scheduling solutions is given by $n!^m$. And the complexity gets even bigger if uncertainty and other features of the problems as setups are considered. Since the original mathematical formulation, developed in the 50s, many types of approaches to formulate adequate schedules has been explored by the literature.

According to Aytug *et al.* (2005), a first classification for these approaches is to split them into two main areas: deterministic and stochastic scheduling research. In deterministic models, all parameters are assumed to be known and the main idea is to plan the work through the machines over a period of time, in the best way possible for optimize a specific objective. Thus, the big majority of the solution methods developed assume that the schedule can be executed in reality exactly as it was conceived (Aytug *et al.*, 2005). However, the

uncertainties inherent to production and the lack of organizational discipline prevent the execution of the theoretical schedule, since it quickly loose quality and feasibility. Besides, most of the algorithms that showed optimal solutions solving academic toy-sized instances, when applied to life-sized instances are very time-consuming (Van Dyke Parunak, 1991). In stochastic scheduling, the input data for the parameters of the problem are not perfectly known and thus they are computed as random variables of which the distributions are known in advance (Bongaerts, 1998). These variations use to represent the *stochasticity* of a manufacturing environment subjected to range of uncertainties such as machine failures, quality problems, arrival of urgent jobs and a myriad of other possibilities. In fact, there are so many specifications of a stochastic problem that there is no framework to fully characterize that class of problems (Leung, 2004).

2.4.2.1 Static vs Dynamic vs Real-Time

Another very common classification of scheduling approaches found in the literature is to distinguish offline/static scheduling, online/dynamic and real time.

The offline scheduling deals with a deterministic problem, since all information is known à priori (Leung, 2004). In this approach, all jobs are received at one specific moment in time and, then, the decision maker has a combinational problem to optimize a predetermined objective. New entering jobs are not admitted until the proceeding scheduling cycle is finished, and thus, a complete understanding of the system and the environment at the time the jobs are available is required. A lot of efficient methods have been evolved to solve this class of problems and many optimal solutions can be found in reasonable time. In the cases that it is not possible to reach the optimal, heuristic methods such as neighbourhood search techniques, metaheuristics or Lagrangian Relaxation, among others have shown good results (Bongaerts, 1998).

However most manufacturing systems operate in dynamic environments in which unpredictable rea-time events may cause a change in schedule plans, turning the previous schedule rapidly infeasible. Usually, dynamic models allow for an intermittently continuous stream of arriving orders that are always included in the current scheduling procedure. In fact, Vieira, Herrmann and Lin (2003) aggregates these events into two categories, making a distinction between situations triggered by resource related issues (machine breakdown, operator illness unavailability or tool failures, etc) and job related issues (rush orders, job cancellation, due date changes, etc).

In a dynamic approach, there is some kind of interactivity involving the scheduler and the real system. In fact, certain events invoke the online scheduler, which generates one appropriate schedule for the actual conditions of the system (Fohler and Fohler, 2015). As stated by Mehta and Uzsoy (1998), Vieira, Herrmann and Lin (2003) and Ouelhadj and Petrovic (2009) dynamic scheduling has been defined under three categories: completely reactive scheduling, predictive-reactive scheduling and robust pro-active scheduling.

In a completely reactive scheduling approach, no base schedule is generated in advance and decisions are made at a local level. For this reason, it is hard to develop global plan for all activities, which makes difficult to predict the system performance. Usually dispatching rules are an intuitive and easy method to apply under these circumstances, but it is important to note that the schedule can be myopic, since that most of the times dispatching rules lack of a global perspective of the system.

In predictive-reactive scheduling, a predictive schedule is generated in advance with the objective of optimizing the performance of the schedule under the estimated conditions. When disruptions occur during production, the predictive schedule is revised in order to maintain its feasibility and improve its expected performance, if possible. In the definition of a predictive-

reactive policy, it is important to take into account two main issues: the definition of the moment of rescheduling and the method that should be used to do it.

Concerning the moment to reschedule, it is possible to revise the schedule continuously, which means every time an event changes the actual state of the system; or periodically, in which rescheduling takes place at predetermined time intervals. Since continuous reschedule generates better realize schedules, but requires higher computation burden, some researchers combine both strategies. This approach involves establishing a periodic rescheduling policy at regular intervals, but if a disturbance with large impact on the system state occurs, the schedule is reviewed (Church and Uzsoy, 1992) .

Regarding the methods to reschedule, four main approaches are mostly used in the literature. In the right-shift rescheduling method, the original sequence of operations is preserved, but the schedule is adjusted moving forward in time the operations to the end of the unexpected event. In complete scheduling, all operations are rescheduled after the trigger. In the match- up rescheduling of Bean *et al.* (1991) the new developed schedule is adjusted to match the predictive schedule at same point later on. Finally, in multi-objective rescheduling, the objective function of the new schedule incorporates not only one performance measure, but also weights of modifications' costs to prevent the new schedule to deviate from de predictive one.

The last reschedule approach is robust scheduling, which attempts to generate predictive schedules that contemplates certain predictability measures to absorb and adjust to some foreseen disruptions. On one hand, the objective is minimizing the lateness to the best schedule solution. And on the other hand, minimize the effect of uncertainties in the schedule, measured by the deviation between the planned and the realised completion time (Ouelhadj and Petrovic, 2009).

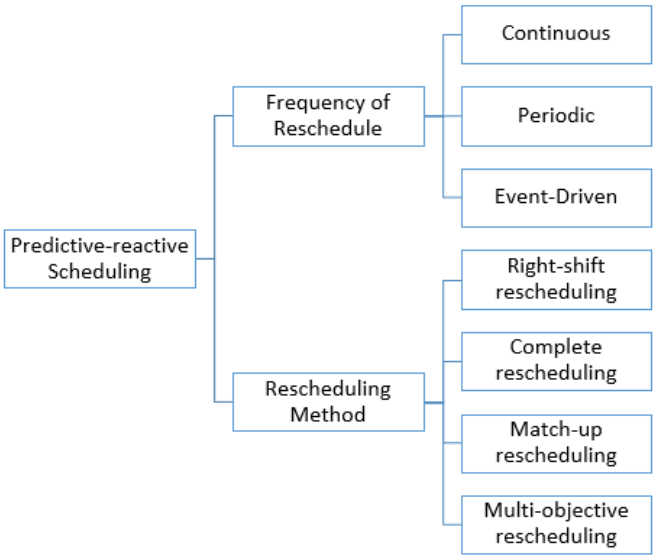

Figure 2: Setting of Predictive Reactive Scheduling

The real-time scheduling is very similar approach to dynamic scheduling. The big difference is that in the online approach, there is no enforcing limits in latency, while in real-time scheduling, the response to the triggered events must be very quick, generally in seconds base.

To conclude, it is noticeable that dynamic and real-time approaches accommodates considerable flexibility and agility to respond to unforeseen system disturbances, but the lack of a global perception, at the end, can generate schedules with performances far from the optimum. On the other hand, a static approach provides a global overview of the system, but

it does not have the capacity to adapt to the unexpected disturbances. Therefore, the behaviour of static approaches is expected to be superior in static environments, while online methods are more likely to better cope with the stochasticity of the system.

2.5 Scheduling methods

This section presents some of the main developments and solution approaches for scheduling problems. A lot of methodologies have been proposed to tackle not only general scheduling instances, but real-world problems as well. The goal is to explain some of the methods that were considered more useful regarding the future direction of the project, clarifying their advantages and disadvantages and the motivation to explore them.

2.5.1 Optimal approaches

For the resolution of basic scheduling problems, optimal methods were proposed for small and some medium size problems. Some of the most known in the literature are presented.

2.5.1.1 Johnson Rule

Johnson's rule was proposed by Johnson (1954), is a method to minimize the makespan in the case of two work centres. Under a couple of conditions, this method prove to generate optimal schedules for the two-machine problem.

2.5.1.2 Dynamic Programming

Dynamic programming is method used to solve complex problems. In a succinct description, the idea behind this technique is to break down the problem in a collection of simpler subproblems recursively, solve them and store all the results to avoid computing them again. Then the dynamic programming algorithm examines the previously solved subproblems and combines their solutions, according to their contribution to the objective function, to find the optimal solution for the global problem (Cormen *et al.*, 2001). At each iteration, it determines the optimal solution for a subproblem, which is larger than all previously solved subproblems.

Held and Karp (1962) proposed well-known equation and a dynamic programming formulation to find optimal sequences either for scheduling problems or travel salesman problem or the assemble-line balancing problem. Authors as Gromicho *et al.* (2012) used that technique to develop more efficient dynamic programming approaches for the job shop scheduling problem. Although this approach presented good results for relatively small benchmark problems, but it can hardly solve more complex problems, becoming impractical to handle real-world problems (Nguyen, Mei and Zhang, 2017).

2.5.1.3 Mixed-Integer Linear Programming (MILP)

The evolution of computer processers enabled to solve some combinatorial problems and Mixed-Integer Linear Programming (MILP) formulations were used to solve small and some medium size instances. Its rigorousness, flexibility and extensive modelling capabilities make it a very attractive methodology to solve some scheduling problems (Floudas and Lin, 2005).

MILP formulations for scheduling can be classified into continuous and discrete time models, regarding its time representation. Discrete splits time into a finite number of periods and each task is associated to one of them. One major disadvantage of this representation is that the solution quality is highly dependent on the size of the periods (short periods increase the solution quality). For that reason, research efforts have been spent to explore continuous-

time models. The model proposed by Manne (1960) and later extended by Liao and You (1996) showed quick optimal solutions for small instances.

2.5.1.4 Branch and Bound

Branch and Bounds procedures are enumeration schemes where certain schedules or classes of schedules are discarded by showing that the values of the objective obtained with schedules from this class are larger than a provable lower bound (Pinedo, 2008). This lower bound is greater or equal to the value of the objective of a schedule obtained earlier. This strategy explores big number of solutions, but prevents an exhaustive enumeration search which would not be viable. To do that Branch and Bound uses a branching rule which states that when the level $k\text{-}1$ are scheduled, one given job j_k only need to be considered if no job still to scheduled cannot be processed before the release time of j_k. That is $r_{jk}<\min_l$ $r_{jk} < min_{l \in J}\{max(t, r_l) + p_l\}$, with J representing the set of jobs not yet scheduled and t is time when j_{k-1} is completed. Figure 3 shows an example of a B&B tree solution for sequence problems

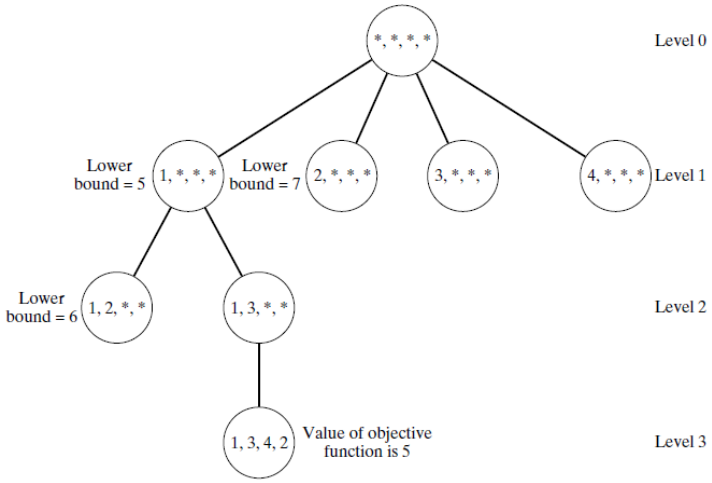

Figure 3: Branch and Bound tree (Pinedo, 2008)

There are several ways to calculate bounds for a node at a given level $k\text{-}1$. An easy method is to use a preemptive EDD rule to schedule the remaining jobs on that node. This rule is known to be optimal to one machine problem with release dates and with the objective of minimizing the lateness ($1|rj,prmp|L_{max}$) and thus it provides a lower bound for the problem.

This method received a considerable amount of attention and reasonably effective enumerative branch-and-bound procedures has been developed. One example is the B&B proposed by Applegate and Cook (1991) which provides a relatively fast optimization procedure for deterministic job-shop makespan problems with size of 10 jobs and 10 machines and 15 jobs and 15 machines. However, when increasing the complexity of the problems, for example by introducing sequence-dependent setup times, the computational time required to solve these algorithms explodes, and they are hardly able to solve the problems.

2.5.1.5 Constraint Programming

Constraint Programming is method evolved used in artificial intelligence, but recently it has been adopted for operations research as an optimization method. In contrast, to mathematical programming, CP does accept nonlinear equations without a large increment of the computational burden.

CP is a method designed to find feasible solutions rather than optimal ones, and for that reason it focuses on the constraints and variables and not in the objective function. The algorithm starts by finding a feasible solution and then a new constraint is created by stating that the value of the objective function must be less (minimization) or higher (maximization) than that of the last solution found. The goal is to gradually narrow down a very large set of possible solutions, ensuring that the objective function improves over time until the optimal value is found.

In the scheduling field of research, authors as Khayat, Langevin and Riopel (2006) applied this approach with good results for relatively small size instances.

2.5.2 Non-optimal approaches

As it was already mentioned, scheduling problems are NP-hard, which do not allow to find always optimal solutions in a reasonable time. Therefore, a lot of research has been done to evolve other methodologies that do not guarantee optimal solutions, but are capable to find reasonable good ones in a relatively short time. In this section, some non-optimal methods are presented, although it should be noticed that there is a vast literature exploring this area and every possible review would not be complete.

2.5.2.1 Constructive and Improvement Heuristics

The complexity of scheduling problems justified the development of heretics, which can be classified into constructive and improvement methods. Constructive heuristics build a schedule starting with an empty solution, and use a repetitive process to construct a complete. In contrast, improvement heuristics develop an existing schedule further by making local moves, as interchanging jobs (Koulamas, 1998). But there are also methods that use a hybrid approach, combining both strategies to evolve good schedules. One of the most popular heuristics for job shop problems is the Shifting Bottleneck heuristic which uses both heuristic methods and it is presented and explained in the section below.

a) Shifting Bottleneck (SB)

One of the most successful heuristic procedures developed for $J|C_{Max}$ is the shifting Bottleneck heuristic. This method was first proposed by Adams, Balas and Zawack (1988) and tries to decompose the problem into a single-machine for each machine at a time. At each iteration, a lower bound is calculated for each unscheduled machine, and the one with the largest lower bound is scheduled optimally without regard to the other unscheduled machines. In the end of this procedure, an attempt to improve the schedules already established is made, before finding the next bottleneck and repeat the process described.

Usually this heuristic uses the disjunctive graph representation, where nodes are tasks to be performed, the conjunctive arcs connect the operations of the same jobs and the disjunctive arcs connect operations processed on the same resource. When the schedule of the critical machine is selected, it is optimized by setting its disjunctive arcs. Figure 4 exhibits an example of disjunctive graph representation.

Lawrence and Sewell (1997) in their research applied this method to 53 deterministic instances, obtaining an average of the ratio primal/dual gap of 4.4%. In their SB heuristic they use a branch and bound method to solve the single-machine subproblems generated

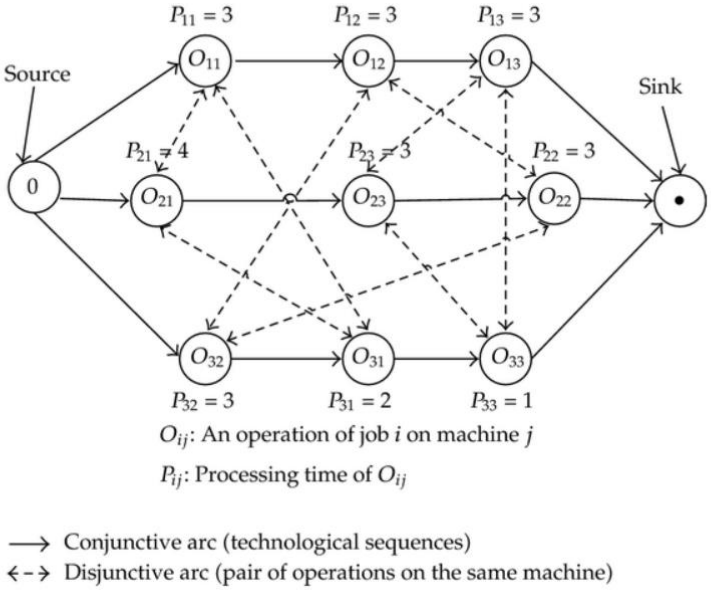

Figure 4: Disjunctive graph representation (Barzegar, Motameni and Bozorgi, 2012)

2.5.2.2 Meta-Heuristics

Meta-heuristics are high level heuristics which guide local search methods to escape from local optima, exploring a larger solution space (Hilier and Lieberman, 2015). In all metaheuristic procedures, it is critical to find a right balance between diversification (exploration of the search space, getting out of the local optimum) and intensification (exploitation of the accumulated search experience, usually ending in a local optimum) (Blum and Roli, 2003).

There are many ways to classify metaheuristics algorithms. One of the most common to distinguish these methods is characterizing in population-based or single point search, depending on the number of solutions used at the same time. Algorithms that work with single solutions, also called trajectory methods, use to comprehend local search-based metaheuristics, like Tabu Search. In contrast, Population-based metaheuristics, such as Genetic Algorithms, describe the evolution of a set of points in the search space.

Due to its adaptability, these methods and capacity to find reasonable good solutions for very complex problems, metaheuristics have attracted the attention of many researchers. Tabu search , simulated annealing, and genetic algorithms have been frequently used to solve static deterministic production scheduling problems (Ouelhadj and Petrovic, 2009). However, in dynamic scheduling, less attention has been done to this class of methods, since usually they require a relatively big computational burden. When they are applied to dynamic problems, the approach involves the decomposition of the problem into corresponding static scheduling sub-problems and then use meta-heuristics to solve those static problems (Lou *et al.*, 2012).

a) Genetic Algorithms (GA)

Genetic Algorithm is a population-based metaheuristic based on an analogy to a natural phenomenon of the theory of evolution formulated by Charles Darwin, firstly proposed by Goldberg, Korb and Deb (1989).

For each iteration of a GA considers a population of solutions, which represent the currently living members of the species. Then, a portion of those members is selected to breed a new generation. Usually, that selection process is biased towards the choice of the fittest members of the population. To generate a second population of solutions, the selected members are combined through genetic operators, namely crossover and mutation. Crossover

involves selecting a pair of "parents" solutions to produce "child" solutions, who share some of the features of both "parents". Since the fittest members are more likely to become parents, the genetic algorithm will gradually tend to generate fitter populations. Sporadically, mutation may occur, this means that certain children may possess features that are not owned by either parent. This operator enables the GA to explore other regions of the solution space. Finally, the process described so far is repeated until some termination condition is reached.

Although these algorithms seem more suitable to solve static deterministic problems because of the high computational time spent, some authors as Chryssolouris and Subramaniam (2001) developed genetic algorithm for dynamic scheduling problems. In fact, he obtained significantly superior results over common dispatching rules for medium size instances with a stochastic environment.

2.5.2.3 Dispatching Rules

In dynamic scheduling problems, especially when decisions must be made in a short time frame like in real-time scheduling, there is the need of methods that provide quickly relatively good solutions. For that reason, dispatching rules have been extensively applied in research and practice (Nguyen *et al.*, 2012).

Generally dispatching rules are greedy heuristics that assign a priority to all available operations to be scheduled, and the one with highest priority is selected to be processed. This type of rule considers the sequencing decision as a set of independent decentralized one-machine problem (Haupt, 1989). Since they can be easily modified when real-world characteristics of the system change and they are easily scalable to every problem size, dispatching rules has been widely explored both by researches and practitioners (Nguyen *et al.*, 2012).

In the work of Lawrence and Sewell (1997) several dispatching rules are tested in 53 instances, considering several uncertainty levels in processing times. The authors compared the performance of the rules between each other, with other heuristic and with the optimal results. The rules tested are briefly explained in this section.

First-come, First-Served (FCFS): Tasks are sequenced according to their release dates. This rule aims at reducing the variation of the waiting time between all the different tasks.

Short Process Time / Longest Process Time (SPT) / (LPT): Tasks are sequenced according their processing time. SPT is used to minimize the mean job flow time, at the expense of jobs with long processing times. LPT is used to balance the load in problems with parallel machines, because at the end of the time horizon tasks with shorted processing time can be used to adjust some gaps created by larger tasks.

Largest Successive Difference (LSD): Tasks are sequenced according to the difference between the processing time on its successor machine and the processing time of the current machine. Whenever a machine becomes available, the job in queue with the larges difference is selected. This rule aims at reducing machine starving

Most Operations Remaining (MOR): Tasks are scheduled according to the number of machines yet to visit. Usually time-in-queue is critical for the job's flowtime and this rule aims at reducing that time by anticipating longer total queue-waits jobs.

Most Work Following (MWF): Tasks are scheduled according the expected processing time on their remaining successor machines and the one with the longest time is selected. Its objective is to minimize the maximum tardiness and lateness.

Longest Tail Remaining (LTR): Tasks are scheduled according the right tail on the distribution of their total remaining processing time. Since Jobs with long right tails have the potential of increasing makespan more than do jobs with short right tails, they are scheduled first. The right tail is calculated as the um of the remaining expected processing time, including the processing time of the current machine, plus a constant multiple of the standard deviation of the remaining processing time, $RT = \sum_{R} pi + K\sigma$, where R is the set of remaining uncompleted operations on the routing job, σ is the standard deviation o its remaining processing times, given by $\sigma = \sqrt{\sum_{R} \sigma_K}$, where σ_K is the standard deviation of the processing time of the job on machine k, and K is a constant.

a) Improved Dispatching Rules

As stated by Pinedo (2008), dispatching rules are clearly a useful methods to find reasonable good schedules to a single objective, but a realistic objective may be a combination of several basic objectives and, sorting the jobs accordingly to only one parameter may not yield acceptable schedules. Also, Blackstone, Phillips and Hogg (1982) refer that there is no single rule that outperforms all the others. In fact, depending on the job configuration, operation conditions and objective function, DR can have large variations in terms of performance. For those reasons, many researchers developed methods to elaborate more effective and adaptive dispatching rules.

i. Combination of Dispatching Rules

One of the most straightforward ways is to improve the performance of DR without affecting their simplicity is to use a combination of simple DRs (Nguyen *et al.*, 2012). Usually researchers use weights to model the rule in conformance to the objectives of the problem (Panwalkar and Iskander, 1977).

ii. Composite Dispatching Rules

Composite dispatch rules are heuristic combinations of single dispatching rules in the form of a sophisticated priority function of various attributes associated either to a job or a machine (Nguyen *et al.*, 2012). Examples of job attributes are weight, processing time and due date, while machine attributes are speed, the number of jobs waiting for processing and the total amount of processing that is waiting in queue. According to Ho and Tay (2005), results show that with careful combination, the composite dispatching rules perform better than the single ones with regards to the quality of schedules.

Modelling composite rules can be a tedious and time-consuming process. Hence, Hyper-heuristics (HH) have emerged as a way to automate the design of heuristics (Waschneck *et al.*, 2016). HH is a methodology to generate heuristics to solve hard computational problems , this means that HH is a high-level approach that given a particular problem instance and an number of low-level heuristics can select the appropriate low-level heuristic (Gendreau, Michel and Potvin, 2010). Usually evolutionary algorithms are employed as HH are employed to evolve dispatching rules. One of those methods that has been successfully applied is genetic programming, which is explained in the next section (Ho and Tay, 2005), (Hildebrandt, Heger and Scholz-Reiter, 2010).

iii. Genetic Programming (GP)

Designing a good heuristic is not a trivial task and, effectively it can be a very time-consuming process and require a great knowledge about the problem. For this reason, Hyper-heuristic methods have been developed to improve the exploration of the heuristic search space. In the last decade, genetic programming has been the dominating technique for designing production scheduling heuristics, due to the flexibility of its representation system and to its powerful search mechanism (Nguyen, Mei and Zhang, 2017).

GP is an evolutionary computation method, based on the Darwinian principle of reproduction and survival of the fittest, firstly proposed by Koza (1992). Fundamentally, GP extends the representation scheme of genetic algorithms into general, tree hierarchical computer programs of dynamically changing size and shape (Miyashita, 2000). At each iteration, GP transforms populations of programs into other population of fitter computer programs. The fitness of the program is determined based on the quality and efficiency of the program in solving the target one. The Figure 5 show how the reproduction process of computer programs is done through crossover.

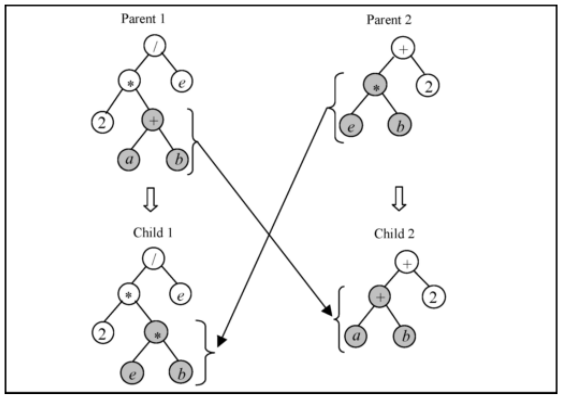

Figure 5: Crossover process of a mathematical expression (Geiger, Uzsoy and Aytuğ, 2006)

Generally, a GP individual is a specific combination of elements selected from two sets. The terminal set, which consist either by program's inputs or constants, and the function set that define the grammar based of GP, and it can include arithmetic operators, logical operators or even specialized functions (Nguyen, Mei and Zhang, 2017).

Figure 6 is a graphical representation of a solution generated by a GP.

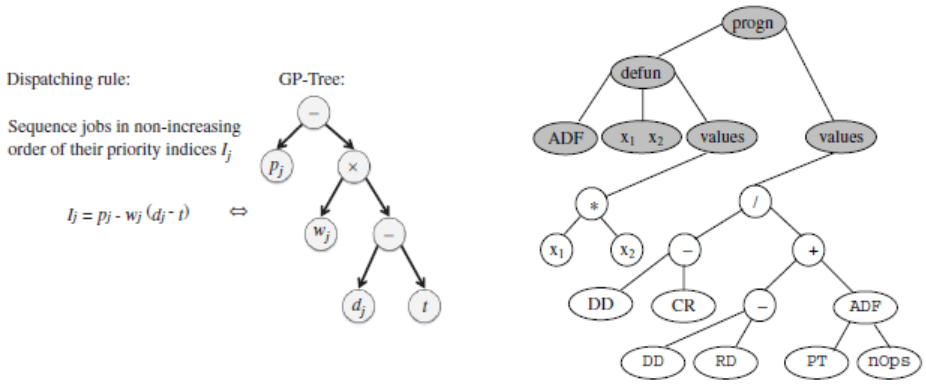

Figure 6: Tree Hierarchical representation of a computer program generated by a GP (Ho and Tay, 2005)

Typically applications of GP are the automatic creation of mathematical formula, but the generation of composite dispatching rules is a problem of a similar nature (Pickardt *et al.*, 2013). Therefore, many researchers as Nguyen *et al.* (2012), Hildebrandt, Heger and Scholz-Reiter (2010), Tay and Ho (2008) Pickardt and Branke (2012) developed GP methods to evolve dispatching rules, obtaining good results both in deterministic and stochastic environments.

2.6 Simulation

Simulation is a technique that aims at virtually reproducing a real-world process or system, imitating its evolution over a time horizon. It uses mathematical models to recreate situations often repeatedly. Usually it is used to analyse stochastic systems that operate indefinitely in various areas of activity such as manufacturing, services, defence and healthcare, among others.

Effectively, the advances of technology contributed to increase the computational speed, and now large time periods can be simulated in a matter of seconds. Since the computer records the performance of the simulated system for a big number of alternative designs or operating procedures, simulation enables to evaluate and compare each alternative, before choosing the most adequate (Hilier and Lieberman, 2015).

Undoubtedly, simulation provides a more intuitive representation of the real system, and reveals cause-effect relationships that would be very difficult to understand through analytical formulations. For example, in the specific case of scheduling, the impacts of dispatching rules are very difficult to be explained, but simulation allow to test and make experiments comparing the performance of the rules for different scenarios.

Since simulation creates a virtual model of the reality, the acquisition of valid source information about system is a critical issue. Furthermore, every assumptions and approximations should be validated to make sure that the model outcomes are trustable.

However, simulations have also disadvantages when compared to analytical models and other optimization methods. In fact, developing a simulation model can take more time than an analytical one and it is also less fit to determine optimal solutions (Hilier and Lieberman, 2015).

2.6.1 Methods of simulation

To better answer to the large range of applications in simulation, four main methods were developed: Monte Carlo Simulation, System dynamics, Discrete Event Modelling and Agent Based Modelling.

Monte Carlo Simulation was first used in the 40s by scientists for the construction of the atomic bomb. It is numerical experimentation technique, inspired by the gambling casinos, to obtain the statistics of the output variables of a system, given those of the input variables (Jacoboni and Lugli, 2012). In each experiment, the values of the input variables are sampled based on their distributions and the output variables are calculated using the computational model. Then, a representative number of experiments should be performed to compute with a degree of confidence the statistics of the output.

System Dynamics is a method developed in the 50s to understand the nonlinear behaviour of complex systems. In this method, the system is modelled as causally closed structure that defines its own behaviour, through circular causal dependencies.

Discrete Event Modelling was first developed in the 60s and it is a technique that approximate continuous real-world processes with non-continuous events defined by the modeller. The model is specified graphically as a process flowchart, where blocks represent

operations. This type of graphic framework is actually very intuitive explains the reason why this method has been so much used in very different areas of businesses.

Finally, agent based modelling was developed in 2002-2003 to answer new requirements of the systems in some areas of business. Nowadays, with the increasing complexity of the systems, the modeller may not be able to express how the process flow works, but he may have some insights about the individual behaviour of each entity. In agent based modelling, agents are firstly characterized individually and then connected to each other in a specific virtual created environment. Thus, the behaviour of the system as a whole emerges from the interactions of all the agents, exhibiting its intrinsic dynamics. It is also noticeable that Agent Based Modelling permits to enhance the extensibility of simulation, since the creation of an agent is independent of the number of agents created. This modularity allows the modeller to stipulate the amount of same type agents that he intends to be necessary. Figure 7 represents the communication between agents and displays a typical state chart that defines the behaviour of an agent.

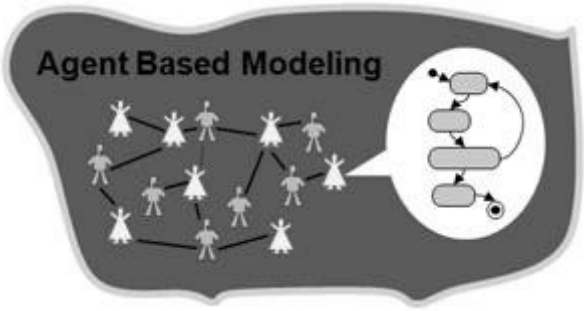

Figure 7: Agent-Based Modelling Scheme (*Anylogic Web Page*, 2017)

2.6.2 Simulation for scheduling problems

Simulation in scheduling is mainly used in dynamic scheduling problems, namely to assess the performance of dispatching rules in the shop floor conditions. In fact, the impacts generated by dispatching procedures are difficult to be explained using analytical techniques and thus, simulation enabled to make rapid progress in this specific field. Kaban, Othman and Rohmah (2012) evaluated and compared the performance of 44 dispatching rules for different measures in a job-shop.

Another application of simulation in scheduling is provided by Kim and Kim (1994), who present a simulation-based real-time methodology for a flexible manufacturing system. In their scheduling approach, they apply priority rules dynamically, based on the experiments tested in a discrete event simulation. The real-time control system reviews the system's state periodically and checks if its performance keeps similar to the simulated one. If the performance of the actual dispatching rule is worse than a given threshold, a new simulation is done for the remaining operations in order to adapt to actual the system's conditions.

Another different application of simulation has been developed in the field of artificial intelligence. Nakasuka and Yoshida (1992) evolved in their research an AI method, using earlier system simulations to determine what the best rule is for each system state.

2.6.3 Simulation-optimization

Traditionally, simulation and optimization are two unrelated concepts of operations research, but the improvement of computational power promoted the design of techniques that combined both (Figueira and Almada-Lobo, 2014). In fact, putting together the great detail of simulation techniques and the ability to find good or optimal solutions with optimization

methods seems to be very a promising approach, though it is not deeply explored by the literature. To provide a better overview of the existing methods in this field, Figueira & Almada-Lobo (2014) proposed a taxonomy to classify the whole range of methods that have been applied. The authors identified four categories of methods in which is possible to screen any simulation-optimization technique. The first is Evaluation Function (EF) that encompasses iterative procedures which use simulation as an evaluator function, orientating the exploration of better solutions in the search space. The second category, Surrogate Model Construction (SMC), comprises methods that also apply simulation as an evaluator. However, their main goal is not to search the solution space, but to formulate a surrogate model used either to guide the search or to be itself explored. The third category includes methods that use simulation for optimization. More specifically, these methods use simulation as a tool to improve the parameters or extend a given analytical model, making it more accurate and far-reaching to different scenarios. Finally, the fourth category, Solution Generation (SG) comprehends methods which use simulation models that incorporate optimization procedures to generate solutions.

Despite the evident potential advantages of these methodologies, they have not attracted a lot of attention in the literature to solve scheduling problems. The approaches found involve most of the times the use of simulation to evaluate simple scheduling rules and select the one which shows the best performance regarding a determined objective (Kim and Kim, 1994). Instead, one possible interesting use of simulation could be to use it as Evaluation Function to evolve simple rules.

2.7 Relevant techniques for this research

In the scope of this research, agile decentralized methods are evaluated for a job shop problem. Hence, from the methods above presented, this research explores dispatching rules, constructive heuristics and meta-heuristics for the scheduling problems. Furthermore, simulation and simulation optimization techniques are used both for evaluating scheduling methods and for evolving a new adaptive composite dispatching rule.

3 Description of case studies

The research conducted is based on by two case-studies, namely the work of Lawrence and Sewell (1997) (L&S) and a real-world case of a Portuguese company of metal packages. In this section both cases are presented and explained.

3.1 Lawrence and Sewell's research

The research of Lawrence and Sewell (1997) studies the performance of static and dynamic methods to solve job-shop problems. More specifically, these authors examine the trade-off between fixed schedules and dynamic schedules, solving problems with uncertainty at processing times' level. Moreover, optimal algorithms and heuristic techniques are applied to analyse the relative performance and their robustness to cope with uncertainty.

In their work, L&S evaluated various scheduling methods in 53 standard job shop instances. Makespan is the only objective and the instances range in size from 6 machines and 6 jobs to 15 machines and 20 jobs. The problem set is composed by 5 instances generated by Adams, Balas and Zawack, (1988), 3 instances of Muth and Thompson (1963), 40 instances created by Lawrence (1984) and 5 instances of Applegate and Cook (1991). Table 1 summarizes the size of these instances.

Table 1: Size of the instances studied in the paper

Size (NJob x NMac)	N Operations	N Problems
6 x 6	36	1
10 x 5	50	5
15 x 5	75	5
20 x 5	100	5
10 x 10	100	12
15 x10	150	5
20 x 10	200	5
30 x 10	300	5
15 x 15	225	5
20 x 15	300	2
Total	-	53

Regarding the solution techniques evaluated, L&S tested three types of methods: one optimal solution algorithm, one heuristic method and a few dispatching rules. The optimal solution algorithm (OPT) of Applegate and Cook (1991) is a branch and bound method that uses a shifting bottleneck heuristic to provide good upper bounds and efficiently trim the research tree. This method provided optimal solutions for 42 of the 53 problems and the

remaining were solved with an heuristic proposed also by Applegate and Cook (1991), which provided results with an average primal/dual gap of 3,5%. The heuristic method tested was the shifting bottleneck heuristic (SB), firstly proposed by Adams, Balas and Zawack (1988). Furthermore, seven dispatching rules were assessed: First-Come first-served (FCFS), Shortest processing time (SPT), Longest processing time (LPT), Largest successive difference (LSD), Longest tail remaining, (LTR), Most work following (MWF) and Most operations remaining (MOR).

For the experiments of static schedules, all the methodologies were evaluated, while for dynamic schedules, optimal solution techniques were not tested. In fact, dynamic sequences have higher computational requirements, and test the optimal solution method would involve too much computational effort.

The stochasticity of scheduling was implemented by introducing an uncertainty component in processing times, considering 10 possible levels of variation.

3.1.1 Results and conclusions from the paper

The principal results of the experiments are summarized in Table 2, which reports the mean makespan performance as a fraction of the optimal makespan of the corresponding problem for all problem instances. The results of the different dispatching rules are exposed for both static and dynamic scheduling approaches.

Table 2: Mean performance of scheduling methods applied in the paper of L&S

	Results					
Rule	0	0,2	0,4	0,6	0,8	1
SB/DYN	1,016	1,056	1,127	1,236	1,329	1,434
MWF/DYN	1,113	1,130	1,169	1,253	1,329	1,423
LTR/DYN	1,137	1,157	1,200	1,285	1,358	1,454
LSD/DYN	1,165	1,186	1,226	1,307	1,380	1,454
MOR/DYN	1,199	1,211	1,249	1,334	1,409	1,494
FCFS/DYN	1,199	1,211	1,249	1,334	1,409	1,494
SPT/DYN	1,215	1,231	1,271	1,343	1,420	1,522
LPT/DYN	1,317	1,340	1,381	1,481	1,557	1,650
OPT/SEQ	1,000	1,080	1,198	1,351	1,487	1,630
SB/STAT	1,037	1,097	1,201	1,346	1,476	1,618
MWF/STAT	1,113	1,155	1,241	1,381	1,500	1,627
LTR/STAT	1,137	1,185	1,274	1,412	1,532	1,665
LSD/STAT	1,165	1,220	1,319	1,468	1,597	1,730
MOR/STAT	1,199	1,237	1,318	1,450	1,567	1,699
FCFS/STAT	1,199	1,237	1,319	1,450	1,567	1,699
SPT/STAT	1,215	1,270	1,375	1,506	1,645	1,787
LPT/STAT	1,317	1,373	1,476	1,623	1,755	1,895

Figure 8 displays the graphical representation of the results reported in Table 2, providing a better perception of the evolution of the deterioration of each scheduling method as the uncertainty level increases.

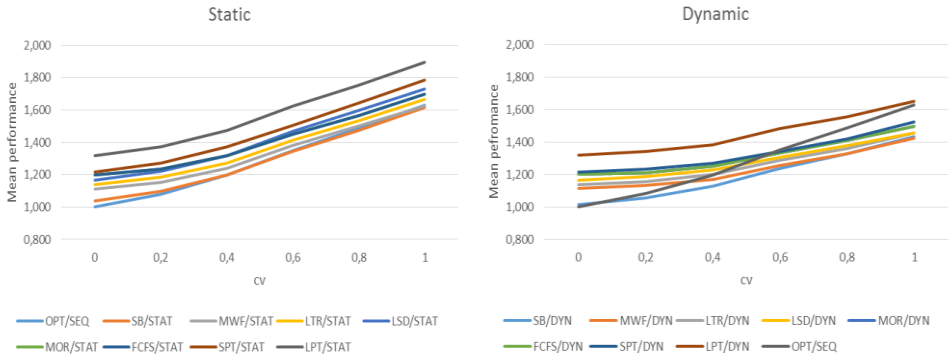

Figure 8: Comparison of the mean makespan performance of fixed (left) and dynamic (right) sequence scheduling methods (based on the research o L&S)

In the analysis of the results of static methods, L&S concluded that, for moderate uncertainty levels, OPT offers a small advantage over SB in exchange for more computational effort. For higher levels of uncertainty, MWF have very similar performance as SB and OPT. It is also worthy to note that for the majority of the uncertainty levels, *SPT* and *LPT* perform worse than the naive *FCFS*.

Concerning the dynamic methods, from the results provided by L&S it can be stated that SB outperforms OPT for nearly any level of processing times' uncertainty. It is also mentioned that *MWF* performs better than the optimal algorithm for high uncertainty levels (cv > 0,3). Actually, its performance converges to SB as uncertainty increases, and MWF requires far less computational effort.

Regarding the comparison of static and scheduling methods, it is evident that dynamic scheduling methods quickly surpass static techniques, even for moderate amounts of uncertainty. In fact, the authors assure that, from a practical perspective, the advantage of more sophisticated solution methods quickly deteriorates as uncertainty increases, and simple scheduling rules dynamically applied can yield comparable or even superior results.

3.2 Real-world case

In order to verify whether the key takeaways from the research of L&S remain valid for more complex problems, the same methodology followed by L&S was applied to a real-world case, where setup times are a major concern. More specifically, the same procedure was followed in the case of a Portuguese company that produces metal packages for different types of products.

The production plant of the company is composed by seven machines which are fairly flexible. In truth, they can perform similar tasks, although with different performances. The jobs that arrive to the system have one or more routes with a certain number of operations associated. The setup times are sequence dependent, which means that the preparation time depends both on the difference from one operation to the next and on the direction of the change. Thus, changing from the operation 1 to 2 takes a different time than changing from to 2 to1, which generates an asymmetric setup matrix. It is also important to highlight that setup times have a large relative proportional representation to the processing times, and thereby they constitute a major difference from the problems investigated in the research of L&S.

According to the literature this problem can be classified as a multi-route job shop, which contrasts with the type of problem studied by L&S. Therefore, in the scope of this research, the routes of the jobs were previously defined, which means that the allocation of the jobs is previously established. With this assumption, the problem can be characterized as a job-shop and compared to the instances evaluated by L&S.

Under these circumstances, 5 instances of the real-world case were generated to serve as a test-bed. Since these problems are real-world examples, they involve a bigger number of orders and operations than those investigated by L&S, which increases the complexity of the scheduling. Table 3 displays the size of these instances.

Table 3: Size of the real-world case's instances

Size (NJob x NMac)	N Operations	N Problems
248 x 6	450	1
239 x 6	416	1
247 x 6	442	1
233 x 6	366	1
238 x 6	401	1
Total	-	5

3.3 Description of the problem under study

In this section, it is presented the characterization and classification of problem and a succinct explanation of the approach followed is provided.

3.3.1 Problem definition

As stated in the subsection 1.3, this master's project aims at extending the research of Lawrence and Sewell (1997) and verify whether their results remain valid for more complex real-world problems, namely with the introduction of sequence-dependent setup times. Thereby in a first approach, a simulation model was developed and validated by comparing the results of the 53 instances studied in the paper.

Then, the first set of experiments was done, testing the more complex environments of instances from the real-world case. In these tests, more advanced setup-oriented methods are implemented both statically and dynamically to evaluate their performance and examine whether they are able to achieve better results. At the end, the results are compared with the set of 53 instances to understand if the conclusions of the paper of L&S remain valid in a different type of problems.

The second set of tests aimed at analysing the effect of sequence-dependent setup times in the performance of agile methods in order to better understand the difference from the results exposed in the paper of L&S. For these experiments, four levels of sequence-dependent setup times were generated and evaluated for the set of 53 problems studied in L&S's paper. In these experiments only dynamic sequences are tested, since the effect of using static or dynamic methods was studied at the first experiments' set. In this study, the objective is to perceive the evolution of the scheduling methods' performance with the gradual increment of setup times.

3.3.2 Characterization of the scheduling problem

The developed simulation model replicates a shop floor in which N parallel machines process M orders. This simulation aims at studying not only the instances of the paper of L&S, but also more complex cases with a higher number of orders and with sequence dependent setup times.

According the classification explained in the Subsection 2.4.1, the machine environment (α) is a typical job shop, since there are N parallel machines and M Jobs with a fixed machine route. The jobs have go through the machines, but not necessarily through all.

The job characteristics (β) of the problems are slightly different considering the 53 instances of the paper and the instances of the real-world case. In fact, none of the instances consider precedence constraints nor pre-emption, and all the jobs are available to right in the beginning. However, they differ from each other in two aspects. Firstly, in the real-world case instances there is sequence-dependent setup time and secondly, they consider recirculation.

Finally, regarding the optimality criteria (γ), all the scheduling methods were evaluated according to two optimization criteria. In the paper of L&S, only makespan is evaluated, but it can be a myopic measure. More specifically, optimizing the schedule of one machine does not guarantee that either the other machines are optimized, or they are being used efficiently. This is especially critical when the workloads of the resources are very distinct from each other.

Therefore, to better assess the performance of the scheduling methods, another metric was considered in the second set of experiments. It was computed the production capacity, which refers to the percentage of time that the machines are effectively working in a specified time horizon, and they are not idle or in a setup phase. In this research, the period considered was the minimum time among the most optimist schedules' times of each machine. This optimist schedule is computed by summing up all the process time of the operations in the machine and adding the minimum setup time of each operation to be performed.

4 Model and experiments' design

This chapter presents the developed simulation model, containing a description of its main components and then an explanation about the functional mechanism that supports the interactions between all the actors involved. Subsequently, the scenarios evaluated in the experiments simulation are described. Finally, the methods to improve and evolved good scheduling solutions are explained.

4.1 Simulation model

The simulation model was developed in Anylogic software which supports object-oriented model design. Its inherent Java environment provides limitless extensibility through the access to external libraries and data sources or by allowing the development of customized Java code. Thereby, all the methods implemented in this model were programmed using Java code.

4.1.1 Selection of the simulation method

The increasing complexity of new manufacturing systems is making scheduling a much harder and elaborated task. Thereby, it is not easy to characterize the behaviour of the entire system, and hardly the impacts of any modification are predictable. In this context, simulation seems an appropriate method to virtually represent the real system, perceiving its cause-effect relationships that otherwise would be very difficult to understand.

Therefore, a simulation model was developed to recreate the typical manufacturing environment of a job-shop problem. For the conception of this model, it was important not only to guarantee that it could represent correctly the complexity inherent to the production system, but also to assure that the simulation model was modular and easy to be extended or modified according to the specifications of on any studied system. This last requirement was imperative, since the objective of this model was to investigate problems with distinct characteristics.

For these reasons, the simulation modelling method selected was Agent Based Modelling. In fact, it is an adequate method both to represent complex flows with various interactions and to enhancing the modularity and extensibility of the model.

4.1.2 Model components

The structure that supports the simulation model is composed by three agents and their interactions. The characteristics of each agent and its behaviour as well as the functional mechanism of its interactions are presented and explained the following sections.

4.1.2.1 Order Agent

The order agent represents one job that arrives at the plant. It is defined by four parameters that must be established at the moment of the creation of the agent:

Id: it is an identification number that allows to distinguish one given job;

Route Machines: A parameter that saves the sequence of machines where the job must be processed;

Route Operations: A parameter that refers to the operations to be processed in each machine;

Route Times: It is the parameter that defines the expected processing times of each operation on the machine.

The order agent can assume two states in the period of time it is in the system. When an order is created it goes immediately to the Waiting state. Then, every time the job is being processed on a given machine it turns to the state InMachine and recovers the Waiting State when the operation is finished. This process is repeated until no more operations are left to perform and then the job reaches its final state. Figure 9 shows the graphical representation of the behaviour of the agent order.

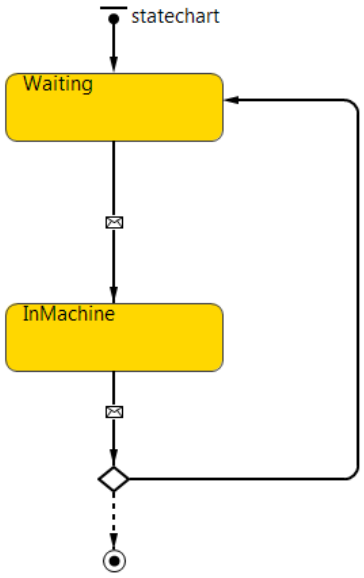

Figure 9: State chart of the order agent

Finally, the variables *processingTime*, *machine*, *operation* and *taskNumber* are used to save the information of the current or next operation to be performed in a given machine.

4.1.2.2 Machine Agent

The machine agent represents the resource machine in the plant. At the moment of creation of a machine agent, the modeller should define three parameters.

Id: It is the parameter that identifies one machine among the others

X: It is the parameter that defines the x coordinate of the position of the machine in the environment

Y: It is the parameter that defines the y coordinate of the position of the machine in environment

The machine agent can adopt one of three possible states while it is in the system. When it is created, it automatically gets Idle, which means that there is no job to be performed at a certain moment. When a job arrives and it is the next to be processed, the machine's state changes to Setup while the machine is being prepared to execute the operation. When the setup is completed, the job starts to be processed and the machine's state changes to Processing. This process is unleashed every time that new operations appear to be processed on the machine. The diagram of the Figure 10 illustrates the state chart of the machine agent

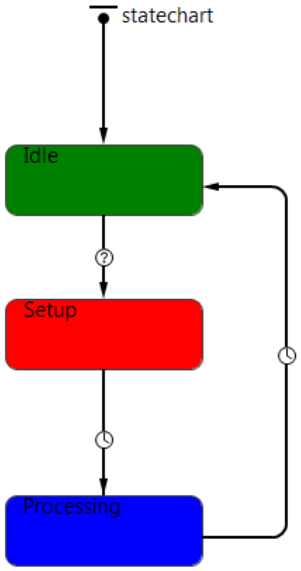

Figure 10: State chart of the machine agent

The machine agent contains variables which store the orders that are being or were previously processed. Another important record is the collection queue which serves as a ledger of the orders that are waiting to be processed on the machine at a certain moment of time. The machine agent is also endowed with functions to calculate the setup time of the operations. Since setups can be dependent on the sequence of operations, every time that an order is going to be processed, the machine immediately computes the setup time of the operation, given the previous processed one.

4.1.2.3 Main Agent

The main agent represents the environment where the other agents of the system interact. Thereby, it is in the main agent that both orders and machines are created and positioned. Then, from the communication between orders and machines and the dynamics of their intrinsic behaviour, emerges the simulation model that can be visualized and recorded.

In the simulation developed within this project, all the machines are created at the start-up and placed in their proper places. Then, an event is automatically triggered at time 0 and all the orders of the problem are generated and sent to the respective machines where they are going to be processed first.

During the simulation, the main agent monitors the number of orders that are already finished and records the finishing times in a dataset. In this manner, when the last order is performed, the simulation automatically points out the value of the makespan.

Figure 11 shows an example of the simulation model provided by the main agent, in which the rectangles are the machines and the circles are the orders. The different colours of the agents represent their state at that moment.

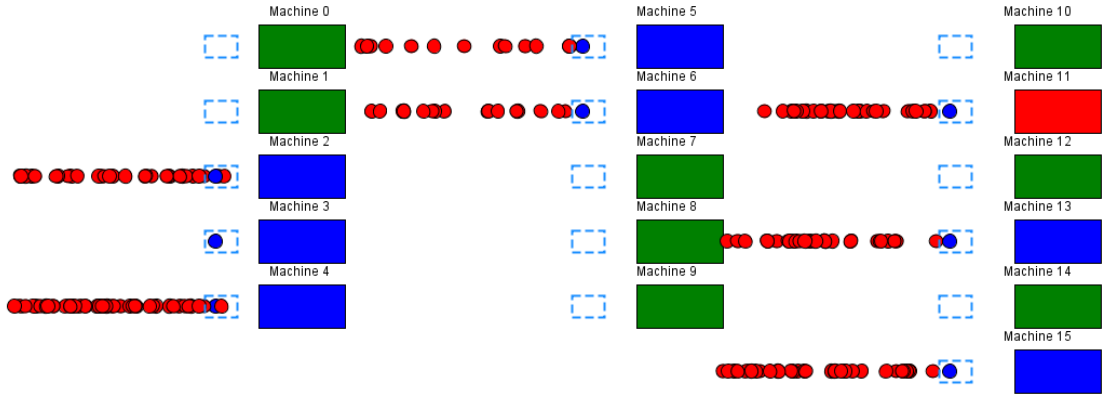

Figure 11: Environment of main agent during the simulation

It is also in the main agent that the settings of the environment are established, namely the unit of time or the speed of the movement. Moreover, the main agent is responsible to receive all the input data from the instances before the simulation starts. For this purpose, the modeller should define before the simulation starts, the parameters of the instance that should be read.

4.1.2.4 Functional Mechanism and Interactions

The functional mechanism behind the simulation model is composed by all the interaction in the system that produce the dynamics of the simulation. Those interactions are a product of the communication between the agents and are regulated by their intrinsic behaviour.

At the beginning of the simulation, all the orders are generated by the main agent, according to the inputs of the model, and then it sends them to the queues of the machines. At this point, the first interaction between the agents in the system occurs, resulting from the communication of the main agent and the machines. In this process, the main agent sends a message with a type order to the machine, which saves that record in its queue collection.

When the queue collection is not empty, the machine agent leaves the state idle and sends a notification message to the next order that is going to be processed. The order changes its state, and sends back a notification to the machine, informing that it is ready to be processed.

When the machine finishes that operation, the order receives a message informing that it is free and should decide whether to go to the queue of the next machine or to leave the system if there are no more operations remaining. The machine changes back to state Idle and picks the next order when it arrives to the queue.

This process occurs in a loop to every order and machine in the simulation until no more orders exist with operations to be performed. The diagram of the Figure 12 illustrates the communication process that was described.

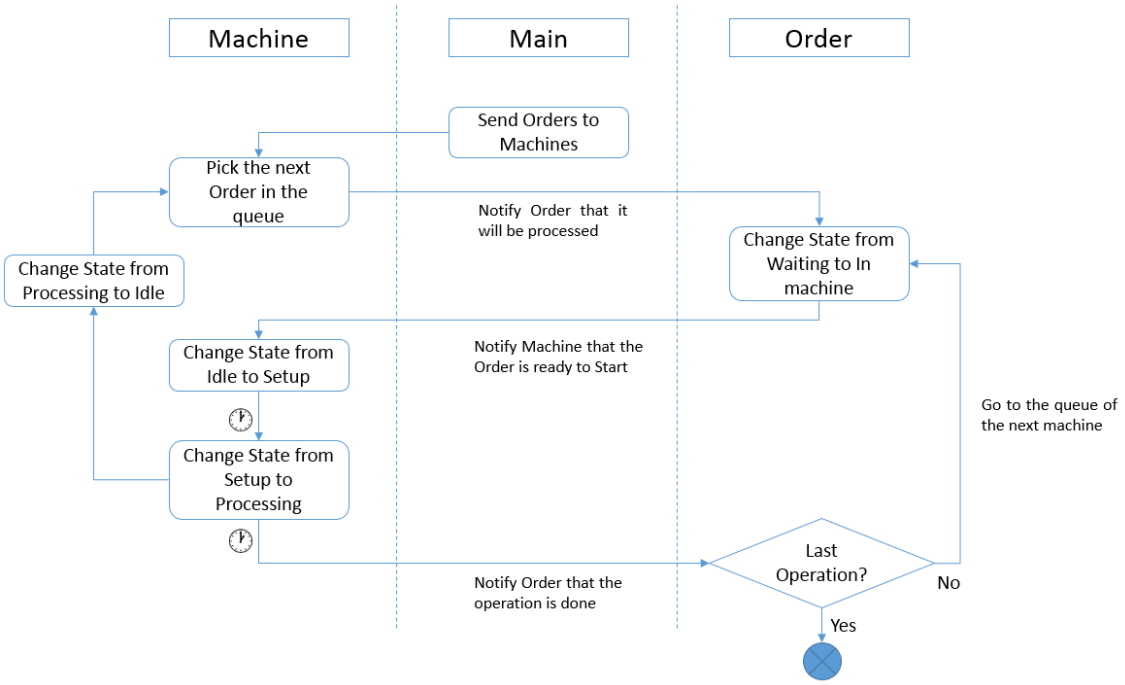

Figure 12: Diagram of the functional mechanism of the interaction between agents

4.2 Experiments' design

The first step of the methodology proposed in this master's project was to develop the simulation model that could replicate the research study of L&S. The model would be validated if the obtained results were similar and then it could be used to study the real-world problem and compare its results with the ones of the generated instances. In this context, the simulation model was prepared to receive the input data either from the deterministic problem set of the 53 instances or the 5 real-world instances or the modified 53 instances with sequence-dependent setup times.

4.2.1 Introduction of uncertainty

Uncertainty was introduced into the problems by perturbing job processing times, as it was done in the research of L&S. Thereby, the processing time provided by the instances was treated as the expected processing time, $pt=E[pt]$. The standard deviation was calculated considering a coefficient that controls the level of variance assumed in a given experiment, $\sigma=cv*E[pt]$. The deterministic case, which considers a null deviation, has $cv=0$, while experiments with higher uncertainty levels consider higher values of cv.

The perturbation of the processing times was introduced using a gamma distribution because it only comprehends non-negative values which makes it widely used in the literature to represent processing times for various applications. The parameters α and β of the distribution were computed through their relationship with the expected value, $\mu=E[pt]$ and variance, $\sigma^2=(cv*E[pt])^2$ of the processing times, which means that $\alpha=\mu^2/\sigma^2$ and $\beta=\sigma^2/\mu$. Hence, it is possible to generate the value of the processing time of a given operation by randomly sampling one value form the resulting gamma distribution.

Nevertheless, it is noticeable that L&S reported that they also implemented a log-normal distribution, but the results did not reveal sensitive to that alteration.

In the designing of the experiments were considered 6 levels of processing times' variation, introduced by setting the values of $cv = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. For each instance and a given uncertainty level, 25 perturbed problems were simulated to study a certain

scheduling method, apart from the deterministic case. This does not require a significant level of replications, since it is not affected by the effect of stochasticity. At the end, with the described experimental design, a total of 126 problems were tested to evaluate one scheduling method for a given instance.

4.2.2 Static vs dynamic experiments

One of the objectives of the experiment is to investigate the relative utility of using dynamic methods rather than static. Therefore, the simulation model was adapted to test scheduling methods both when the sequences of orders for each machine is established before the simulation runs and when the sequence of orders is evolving and being determined while the simulation is running.

For the dynamic approach, all the orders are released to the system and sent to the next processing machine. In the queue of each machine they are sorted according to a certain rule or method that updates the sequence every time a new order arrives to the queue. When the order is finally processed to the machine, it goes towards the next one. This processed is repeated in a loop and a sequence of orders in one machine is dynamically generated. It is worthy to note that dynamic sequences involve a higher computational burden than static approaches, since the simulation model must recalculate the sequence of the queue every time a new operation arrives.

For the static approach, fixed-sequence schedules are inputted to the system, which pre-establishes the queue collection of each machine and does not allow any possible modification during the simulation. Thereby, the machine picks an order only when it is available in the queue. Otherwise, the machine must wait until the the order arrives.

In these circumstances, static sequences had to be generated before the simulation runs, for each scheduling method implemented. For this purpose, the methods were firstly simulated in the dynamic model and the sequences resulting from the deterministic case in each machine were recorded. Then these sequences were used as the input of each machine for the static approach. It is noteworthy that assuming equal sequences for static and dynamic approaches in the deterministic case is a valid assumption for all the methods evaluated, since dynamically updating the schedule do not allows for improvement opportunities for any of the evaluated methods.

4.2.3 Lower Bounds

Effectively, comparing directly the resulted makespan of the instances is not a valid approach, since they have different sizes and processing times. Thereby, in the first set of experiments, the same procedure followed in the work of L&S was used to measure the quality of the scheduling methods evaluated. In their procedure, the maximum machine's makespan of a given scheduling method is compared with the optimal schedule's makespan. However, the authors were only able to compute the optimal schedules for 42 of 53 instances and the remaining were solved using a heuristic method developed by Applegate and Cook (1991). As it is not known which problems were solved optimally, in this research, the makespan value reference of each problem was given by Resende and Gonçalves (2013) who provide optimal makespan values for all the 53 simulated instances. These different reference values may cause a small gap in the comparison of results, when validating the model.

Concerning the real-world case, the reference makespan values are obtained using a complex and centralized heuristic. Despite these values are not optimal, they constitute very good solutions for such large and complex problems.

Regarding the second set of experiments, the instances evaluated are new generated problems, which do not have optimal benchmark solutions available. Thus, optimal lower

bounds of each problem were computed, considering the most optimist scenario, where the operations take their minimum setup time plus processing time to be performed.

4.3 Implemented scheduling methods

In the context of this dissertation, the methodologies evaluated by L&S were first developed to serve as validation of the model. Then, more sophisticated techniques were explored to improve the quality of the schedules. In this section, the scheduling methods implemented in the simulation are presented and explained.

4.3.1 Scheduling methods from Lawrence and Sewell

As it was already mentioned in the section 3.1, the research of L&S evaluated three types of methods: one optimal, the Shifting Bottleneck heuristic and several dispatching rules. In the context of this master thesis not all of these techniques were considered relevant and, thus only two of them were focused.

4.3.1.1 Optimal algorithm

The optimal solution algorithm used in the paper of L&S and firstly proposed by Applegate and Cook (1991) is a good approach to solve deterministic problems of small and medium size. However, when the size and complexity of the instances increases, it requires a heavy computational burden and, in various cases, optimal solutions cannot be found. Effectively, this method was only able to solve 42 of the 53 instances evaluated on the investigation of L&S. Since the purpose of this dissertation is to apply the scheduling methods not only to the 53 instances, but also to bigger and more complex real-world problems involving also sequence dependent setup times, this approach did not seem attractive to cope with such complex problems and, hence, it was not implemented.

4.3.1.2 Shifting Bottleneck Heuristic

The scheduling heuristic implemented in the research of L&S was the Shifting Bottleneck Heuristic of Adams, Balas and Zawack (1988). This heuristic is known to outperform various dispatching rules and have near optimal performances when applied to benchmark J||Cmax problems (Ovacik and Uzsoy, 1997). In concordance, it is the heuristic procedure with the best performance in the research L&S.

However, the method implemented uses an exact branch and bound algorithm to solve their single-machine subproblems and the presence of sequence-dependent setup times renders the subproblems intractable (Ovacik and Uzsoy, 1997). Thereby, as the purpose of the research of this master thesis is not to exactly replicate the study of L&S, but to extend it to more complex and real problems, it was developed a genetic-algorithm to obtain approximate solution to the subproblems.

Still, that the use of any heuristic approach that can be applied to the subproblems does not guarantee feasible solutions at intermediate iterations. The fact that the release times and due dates calculated from the network representation of the partial schedules do not capture all the constraints imposed on subsequent iterations by the partial schedule makes impossible to prevent the generation of unfeasible solutions without seriously affecting the computational effort and the quality of those solutions (Ovacik and Uzsoy, 1997). For this reason, the Shifting Bottleneck heuristic was not included in the model.

4.3.1.3 Dispatching rules

Concerning the dispatching rules evaluated by L&S, all those simple the simple heuristics were implanted in the simulation. The implementation was done through the

creation of a function that sorts the all the orders in the queue collection of a given machine. The function ranks the priority of each order according to the rule and it is activated every time a new order arrives to the queue. When the order leaves the queue, there is no need to call the function again because it would not change the current queue's sequence. The Dispatching rules implemented are First-Come first-served (*FCFS*), Shortest processing time (*SPT*), Longest processing time (*LPT*), Largest successive difference (*LSD*), Longest tail remaining, (*LTR*), Most work following (*MWF*) and Most operations remaining (*MOR*). These methods are summarized in Table 4.

Table 4: Dispatching Rules implemented in the simulation model

FCFS	First-Come first-served
SPT	Shortest processing time
LPT	Longest processing time
LSD	Largest successive difference
LTR	Longest tail remaining
MWF	Most work following
MOR	Most operations remaining

4.3.2 Advanced setup-oriented scheduling methods

Besides the methods evaluated in the research of L&S, a meta-heuristic, a scheduling heuristic and a dispatching rule were developed. All these methods have in consideration the setup time, which is a critical factor in more complex environments. Next, these techniques are presented and explained.

4.3.2.1 Genetic algorithm (GA)

The Genetic Algorithm implemented was based on the Biased random-key genetic algorithm proposed by Gonçalves and Resende, (2011) for combinatorial problems. In this technique, the chromosomes are represented as a vector of randomly generated real numbers in the interval [0,1]. Then, a decoder algorithm associates it to a solution of the combinatorial optimization in order to calculate the fitness of the solution. The association is done by sorting the vector of random keys and then the resulting ranking of the sorted keys can represent a sequence. The Figure 13 illustrates the decoding process described.

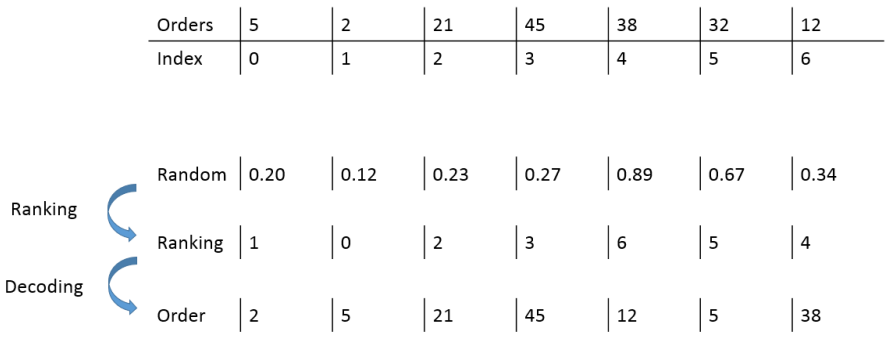

Figure 13: Encoding Process of the genetic algorithm

The GA method implemented generates initially a population of 75 random-key vectors which provide 75 different solutions. After computing its fitness, a tournament selection method choses 20 solutions to produce 50 new solutions created by crossover. The tournament selection choses randomly 2 solutions, compares them and chooses the best with a probability of 75%. This process is done iteratively until 20 random-key vectors are selected. The crossover process picks two random solutions of the group of 20 and computes its fitness. Then a new solution is generated by selecting the elements of the best parent with a probability of 70%. This way, introducing a bias on selection of each element of the new vector, the new generated solutions are more likely to inherit characteristics of the best parent.

At the end, the new produced solutions and the parents are put together and compared. Then half of them (35) will remain in the algorithm, while the 35 missing solutions to continue the algorithm are generated in a new iteration. This process is done in a loop until 100 iterations are performed. Figure 14 shows a flowchart of the described process.

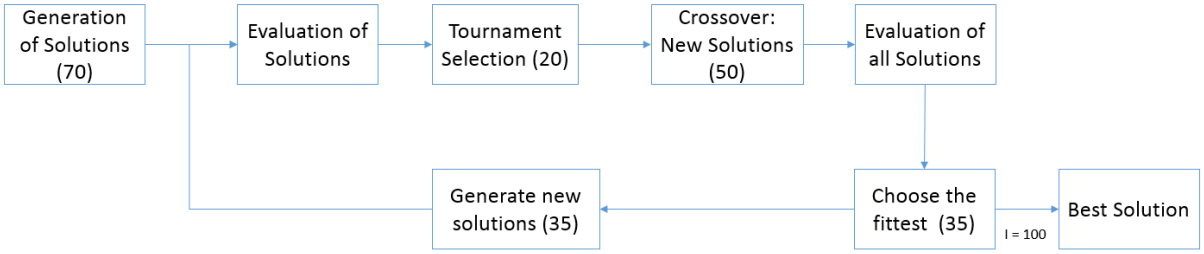

Figure 14: Flowchart of the genetic algorithm implemented algorithm

The parameters of the implemented algorithm were tuned on an experimental basis. The algorithm does not perform an exhaustive search due to the fact that it would become too time-consuming, especially if applied in dynamic schedules. Thus, the parameters were established finding a good balance between the quality of the solutions and the time of running.

4.3.2.2 Greedy heuristic (GH)

The greedy heuristic is a simple priority rule only applicable to instances with setup time. This method sorts the queue by picking, in each evaluation step, the element among those who are available with the least setup time, given the order that is processed in front of it. When setups are sequence-dependent, the setup of each operation changes in each evaluation step and thus the setups of all operations available to be chosen next to have to be recalculated. This heuristic is named “greedy” since it always looks for the best element according to the rule, and for this reason it may yield myopic solutions.

4.3.2.3 Constructive Heuristic (CH)

The constructive heuristic implemented is a scheduling method that aims at extending the capabilities of the greedy heuristic. In fact, choosing always the order with minimum setup may provide a worse solution in the end than if we consider the possibility of choosing not the element with the shortest setup time, but one of the shortest.

Therefore, this heuristic generates a population of solutions with a selection method that enables to select other elements rather than the one with the least setup time be selected. In the end, the queue setup time of each solution is computed and the one with the shortest setup time is selected.

Effectively, the selection method uses a similar approach to the greedy heuristic, regarding the comparison of each element with the one that will be processed in front of it.

However, instead of choosing the shortest setup time, weights are calculated for each element, $Wi = (TS/Si)^f$, in which *Wi* is the weight of the element *i*, *TS* is sum of the setup times of all elements and, *Si* is setup time of the element *i* and *f* is the factor of greediness. Then the probability of selection of each member is computed by normalizing the weights of the elements. Given these probabilities, the next member of the queue is selected. It is noteworthy that the bigger the factor of greediness, the greedier are the methods and more difficulty exits in evolving new solutions. This parameter was defined equal to 3 in the simulation model. In Figure 15, the pseudocode of this method is described.

```
Function Constructive Heuristic (OperationReference, queueOperations)
If not OrderInProcess then
    Select pair (O1, O2) in queueOperations with minimum setup
    InitialList.append(O1,O2)
    queueOperations.remove (O1, O2)
    OperationReference ← O2
Else
    OperationReference ← OrderInProcess
End if
Repeat k times
    List_k.append(initialList)
    availOperation ← queueOperations
    While availOperation > ∅ do
        totalSetup ← 0
        For all op ∈ availOperations do
            setup_i = setup (OperationReference, op)
            totalSetup += setup_i
        End for
        totalSecore ← 0
        For all op ∈ availOperations do
            score_i = (totalSetup / setup_i) ^f actorGreedy
            totalScore += score_i
        End for
        For all op ∈ availOperations do
            prob_i = score_i / totalScore
        End for
        Select op ∈ availOperations according to prob values
        queueSetup_k += setup (operationReference, op)
        operationReference ← op
        list_k.append(operationReference)
        availOperations.remove(OperationReference)
    End While
Select list_k with least queueSetup_k
```

Figure 15: Pseudocode of the Constructive Heuristic scheduling method

4.3.2.4 Short setup and processing time

Setup and Processing Time (*SSPT*) is a dispatching rule proposed by Wilbrecht and Prescott (1969), which can be described as modification of the Shortest Processing Time (*SPT*). Effectively, *SSPT* is a rule that adds the setup of a job to its processing time when determining the priorities of the orders. Then, the order with the smallest priority index is selected to be processing next According to Pickardt and Branke (2012), this rule is more effective in minimizing the flowtimes than either the *SPT* or the *GH*.

4.3.2.5 Minimum Marginal Setup Time (MMS)

Minimum Marginal Setup Time (*MMS*) is dispatching rule, firstly suggested by Arzi and Raviv (1998). This rule is slightly more sophisticated than *GH*, since it considers more operational variables that just the setup time. In fact, in this methods the job's required setup time is divided by the number of waiting operations to determine its priority. Then the job with the shortest marginal setup time is selected to be processed. Thereby, this rule combines two objectives, focusing not only on the orders with a short setup time to improve the mean flow time, but also looks at reducing the time-in-queue by taking into consideration the longer total queue-waits jobs.

4.3.2.6 Shortest Normalised Setup and Processing Time (*SNSPT*)

Shortest Normalised Setup And Processing Time (*SNSPT*), proposed by Kochhar and Morris (1987) is a rule, in which both processing time and setup time are divided by their respective average values in the queue and weighted before being summed up to form the priority index. In fact, this rule enables to adjust the method to the circumstances where it is applied by giving preference to one of the included variables. However, the authors do not provide any guidance on how to derive properly the weights of the rule and, thus, in the case of this project, the weights of processing time and setup time were considered equal.

4.4 Sequence-dependent setup times' generation

In order to create comparable problems, the generation of setup times for the set of 53 instances should consider not only the relative relation between the sizes of the setup times and processing times of the real-word instances, but also the mean processing time of each instance for which it is generated. Thereby, the setups modelling involved two phases. In the first, it was computed the average proportion of setup times and processing times of the real-word instances. Secondly the setup times was generated through the construction of a matrix of setups for each one of the 53 instances studied by L&S.

In the first phase, it was computed the mean processing times and setup times of each real-world instance as well as its standard deviations. Then calculated the mean proportion of the setup times and processing times and the proportion of its standard deviations was calculated. In fact, it was verified that the setup times are relatively large in comparison with the processing time, since the calculated proportion for the times was $p(\frac{ST}{PT}) = 1,1624$ and for the deviations $p(\frac{stdev(ST)}{stdev(PT)}) = 0,2357$.

In the second phase, the mean processing time and its standard deviation were calculated for all the 53 instances. The dispersion of the obtained values indicated that 48 problems have similar mean processing times and standard deviation, 3 of the remaining also share identic values and the others have very distinct mean processing times. Figure 16displays the dispersion of the obtained values.

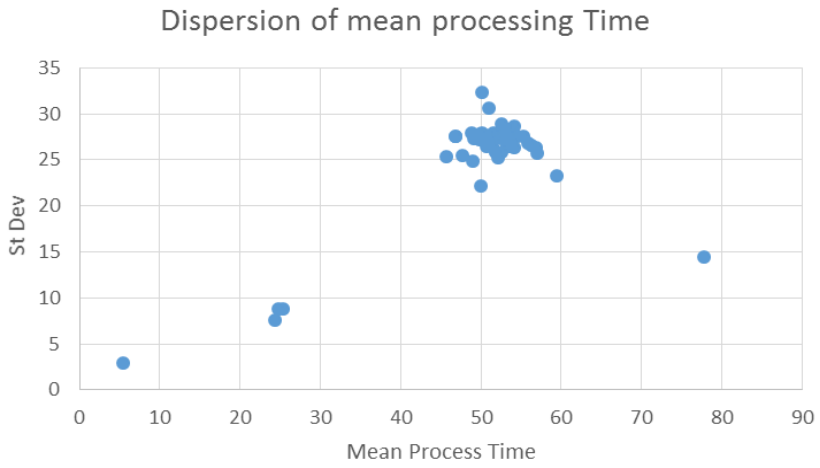

Figure 16: Dispersion of the processing time of the set of 53 instances

Therefore, it was necessary to generate four different distributions to sample the setup times' values for all the instances. It was used a gamma distribution to sample random matrices for each problem.

Since the purpose of this research is to evaluate the effect of the setup times as a differentiator factor of the problems, four setup levels were generated. The first keeps the same proportion of the setup and processing times and the proportion of the standard deviation of the real-world instances. The second level considers the proportions multiplied by a factor of 0,75; the third multiplies these proportions by 0,5; and the fourth by 0,25. In these circumstances, four matrices of setups were generated for each problem, which results in a total of 212 setup times' matrices generated.

4.5 Validation of the simulation model

This section presents the evaluation tests and analysis executed to validate the simulation model. The validation was done by comparing the results of the scheduling methods implemented for the set of 53 instances with the results provided by the research of Lawrence and Sewell (1997). Then, the quality of the results of the simulation and the possible causes behind some differences are discussed.

The principal results of the experiments are summarized in Figure 17, which reports the mean makespan performance as a fraction of the optimal makespan of the corresponding problem for all instances. The top graphics display the results from the research of L&S, and the results obtained are showed in the bottom of the figure. The mean performance is represented as a function of the uncertainty level, by gradually increasing cv.

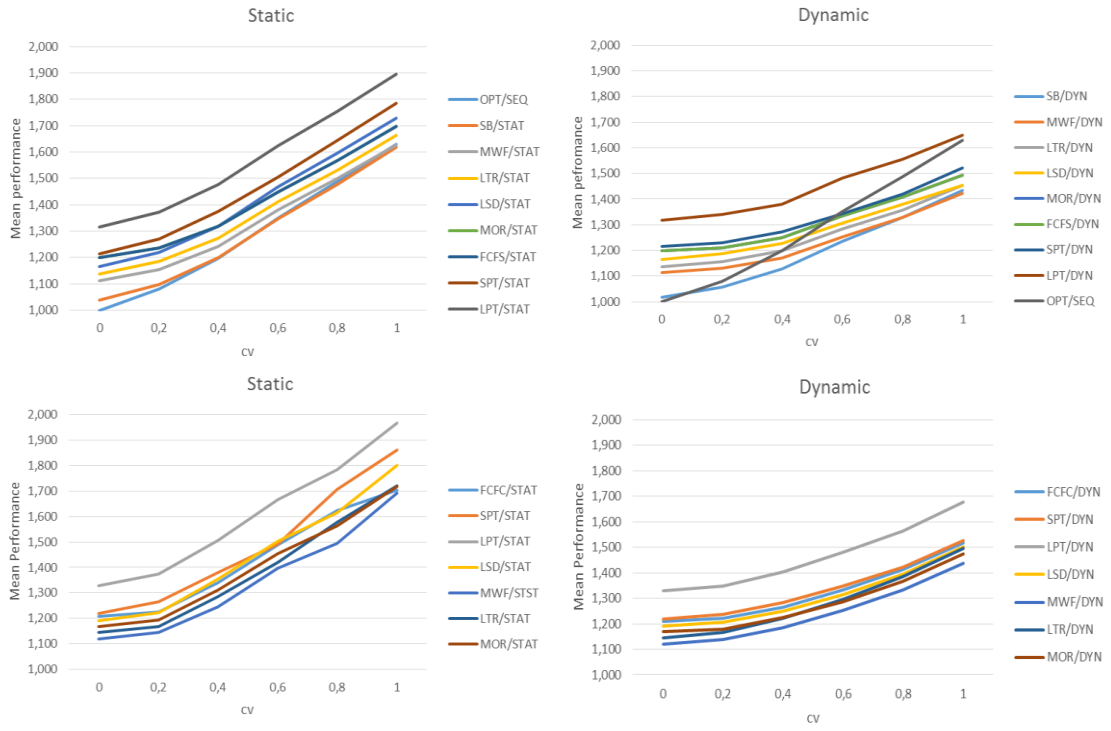

Figure 17: Comparison of the mean makespan performance fixed (left) and dynamic (right) sequence scheduling methods in the research of L&S (top) and in obtained results (bottom).

Analysing the graphics, it is possible to verify that they have a similar shape, which indicates that the simulation is providing similar results. However, the introduction of uncertainty generates different conditions for the problems, and the effect of stochasticity can induce some divergences in the final results.

Therefore, a rigorous comparison between the results from the simulation and those from the research of L&S was made only for the deterministic case to have fixed reference values to compare with. Since for the deterministic case, the results of the methods were equal for static and dynamic approaches, it was only necessary to compare one of the approaches.

The results obtained in the simulation and the results of L&S are put together in Table 5. On the right side of this table, the performance of methods evaluated is ranked.

Table 5: Results obtained for the various scheduling methods and their differences from the paper of L&S

	Differences			Ranking	
Scheduling Method	Results obtained	Paper form L&S	Differences	Results Obtained	Paper form L&S
MWF	1,120	1,113	0,007	1	1
LTR	1,145	1,137	0,008	2	2
LSD	1,191	1,165	0,026	4	3
MOR	1,168	1,199	0,031	3	4
FCFC	1,208	1,199	0,009	5	4
SPT	1,220	1,215	0,005	6	6
LPT	1,328	1,317	0,011	7	7

The analysis of Table 5 suggests that there are slight differences between the results obtained and those from the research of L&S. Although the differences are residual, it is believed they can be explained by two factors. First, as stated in section 4.2.3, the lower

bounds used to compute the fractional value of the performance are, in some cases, smaller than those used by L&S. Thereby, it is expected that the value of the mean performance of the scheduling methods can be marginally bigger than in the results of L&S, which happens for almost all the rules tested except for *MOR*.

The second possible explanation of the differences is the fact that there is no second rule to apply in cases of a draw between the priorities of orders. Since, in these cases, no rule is explicitly applied in the research of L&S, *FCFS* was the decision criteria implemented. However, *FCFS* is a naive rule, which essentially provides random sequences, and thus there may be some differences in the scheduling methods where more draws happen. Effectively, *MOR* is the method more subjected to these situations and it is the one which registers the bigger difference. It is also believed that this fact can be the cause of the incoherency of the fractional performance of *MOR* be bigger in the results of L&S than in the simulated ones.

Finally, it is noteworthy that despite the existence of these small differences, the relative performance difference between the methods remains almost the same. In fact, according to the right side of Table 5, only two rules switch positions in the performance's ranking of the methods.

In conclusion, the differences found between the simulation and the research of L&S are marginal and can be explained. Therefore, the simulation seems not to have external factors of deviation and thereby it is validated.

5 Analysis of results

This chapter presents the simulation results obtained and a discussion about their analysis in light of the research topics of this dissertation. It is divided into two major sections. The first addresses the first set of experiments, where techniques used in the research of L&S and the more advanced setup-oriented methods are evaluated for the real-word instances. The second section describes the results and analysis of the second set of experiments, where the same methods are evaluated for four setup levels generated for the set of 53 instances of the research of L&S.

5.1 First experiments' test

The first experiments' test aims at studying whether the conclusions of the research of L&S remain valid for larger and more complex problems with sequence-setup times. More specifically, in these tests either the relative utility of dynamic methods over static, or the evaluation of the performance of agile scheduling methods, or the quality loss of decentralized methods are studied in a more complex problem. Moreover, more advanced setup-oriented methods are implemented to compare their performance with those of the techniques used by L&S.

In this set of tests, the only optimality criteria used is the makespan and the results are presented as a fraction, in which the nominator is the makespan and the denominator refers to the quasi-optimal solution for the respective problem. Thereby it is possible not only to compare the performance of problems with different sizes and processing times, but also to perceive the loss of quality of agile methods against the centralized ones.

In these experiments, both static and dynamic approaches are tested and compared for all the scheduling methods already mentioned.

5.1.1 Results

Table 6 displays the results obtained for all the scheduling methods for both static and dynamic approaches. At the bottom of the table, the mean performance of the more sophisticated centralized heuristic is exhibited. Each cell refers to the mean performance of all the five instances evaluated.

Table 6: Results obtained for each scheduling method in the first set of experiences

	Results					
Method	0	0,2	0,4	0,6	0,8	1
FCFC/DYN	1,444	1,462	1,459	1,476	1,502	1,532
SPT/DYN	1,501	1,525	1,540	1,552	1,585	1,624
LPT/DYN	1,496	1,515	1,506	1,532	1,541	1,591
LSD/DYN	1,431	1,462	1,443	1,471	1,488	1,512
MWF/DYN	1,443	1,450	1,449	1,465	1,501	1,514
LTR/DYN	1,502	1,509	1,512	1,516	1,552	1,568
MOR/DYN	1,440	1,438	1,434	1,440	1,461	1,493
SSPT/DYN	1,444	1,373	1,378	1,400	1,409	1,444
MMS/DYN	1,012	1,031	1,066	1,102	1,153	1,148
SNSPT/DYN	1,158	1,168	1,203	1,212	1,269	1,306
GA/DYN	1,390	1,431	1,423	1,378	1,470	1,541
GH/DYN	1,047	1,061	1,124	1,106	1,166	1,249
CH/DYN	1,046	1,056	1,080	1,119	1,145	1,239
FCFC/STAT	1,444	1,443	1,446	1,465	1,484	1,563
SPT/STAT	1,501	1,510	1,520	1,559	1,613	1,610
LPT/STAT	1,496	1,532	1,565	1,606	1,660	1,725
LSD/STAT	1,431	1,445	1,466	1,492	1,511	1,570
MWF/STAT	1,443	1,446	1,469	1,480	1,511	1,532
LTR/STAT	1,502	1,529	1,558	1,599	1,646	1,665
MOR/STAT	1,440	1,447	1,442	1,441	1,471	1,497
SSPT/STAT	1,444	1,475	1,468	1,515	1,528	1,596
MMS/STAT	1,012	1,035	1,081	1,109	1,147	1,195
SNSPT/STAT	1,158	1,199	1,212	1,229	1,292	1,350
GA/STAT	1,390	1,476	1,497	1,526	1,546	1,594
GH/STAT	1,046	1,065	1,113	1,149	1,201	1,254
CH/STAT	1,049	1,067	1,106	1,133	1,201	1,295
OPT/STAT	1,000	1,023	1,064	1,099	1,153	1,192

5.1.2 Discussion of the results

The discussion of the results from the first experiments' set aims at providing answers to the three above mentioned main objectives of these tests.

Regarding the study of the relative utility of dynamic over static methods, it is verified that dynamic techniques do not demonstrate a clear advantage over static as L&S present in their research. In fact, the difference between the slope of dynamic techniques and static is considerably lower than the displayed in the results of L&S, which indicates that static methods do not deteriorate so quickly in the real-world case problems in comparison with dynamic methods. This diminishing of the relative utility of dynamic methods can be explained by loss of processing times' proportional importance to influence the makespan, due to the introduction of setup times. In other words, the setup times dilute the importance of processing times as an influence factor of the makespan and thus, their variations are not so explicitly revealed in the final makespan. Figure 18 displays the comparison of the mean performance between static and dynamic methods for both the instances of L&S and those from real-world case.

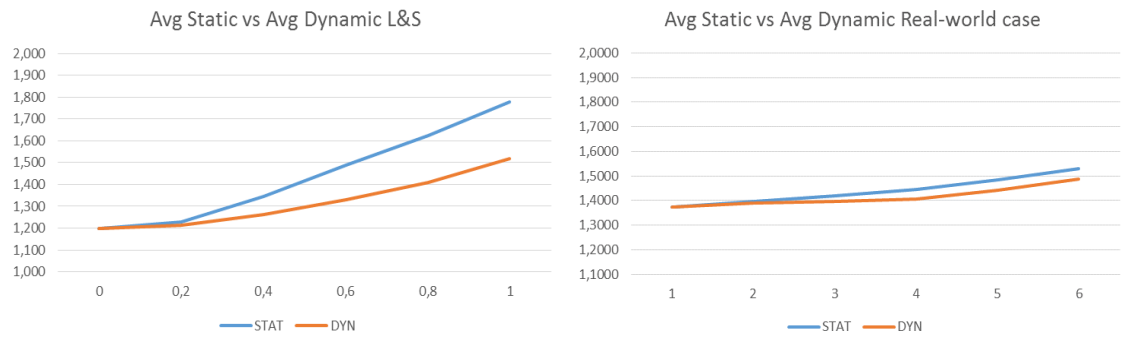

Figure 18: Difference of mean performance of static and dynamic methods for instances of L&S(left) and real-world instances (right)

Concerning the evaluation of the scheduling methods' performance, the results enable to understand that *GH*, *CH*, *MMS and SNSPT* methods have significantly better performance than the other applied rules. In fact, the curve that characterises the performance of these methods is very close to the orange line of Figure 19, which represents the performance of the near-optimal centralized algorithm. This means that local and decentralized setup-oriented methods are able to achieve comparable or even superior results than central and complex algorithms.

These remarks suggest that setup time is a critical factor in problems with sequence-dependent setup times. Figure 19 displays the mean performance of static and dynamic methods for the real-world instances. It shows that *MMS* is the best performing method for almost all uncertainty levels.

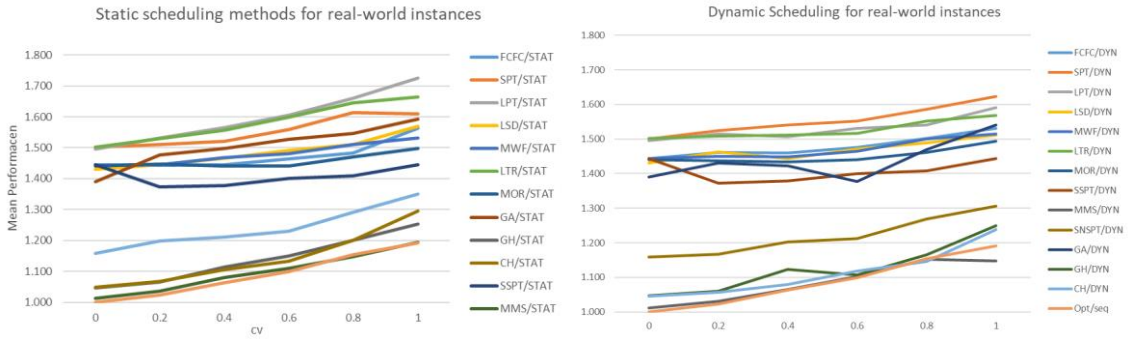

Figure 19: Mean performance of static and dynamic methods for real-world instances

An analysis of the rules that are not setup-oriented enables to verify that the relative performance of the methods seems to maintain quite similar to the results of L&S. In truth, as in the research of L&S, *MOR*, *MWF* and *LSD* are the best performance rules, while *SPT* and *LPT* generate the worst schedules. These results suggest that considering the following work of the jobs, either the number of machines or the amount of processing time remaining, seem to be an important property of a scheduling rule. In contrast considering the current processing time does not benefit the performance of the method. These results can be explained by the fact that uncertainty is introduced through the variation of processing times and thus rules that focus only on the current processing time are more subjected to the effects of variations. Table 7 puts together the mean results from both sets of problems and presents the relative ranking of each method.

Table 7: Mean performance of no setup oriented methods for real-word instances and in L&S research

	Real-world Instances		Instances from L&S	
Method	Mean Performance	Ranking	Mean Performance	Ranking
FCFC	1,479	4	1,384	5
SPT	1,555	7	1,407	6
LPT	1,530	6	1,560	7
LSD	1,468	2	1,372	4
MWF	1,470	3	1,295	1
LTR	1,527	5	1,351	3
MOR	1,451	1	1,350	2

Finally, the first experiments' set aims at investigating the loss of quality of agile and decentralized methods in comparison with central and more sophisticated scheduling solutions. For that purpose, the makespan of the agile methods of each problem is compared with the performance of the complex and centralized algorithm. The deterioration of the solution quality is given by the difference between the fractional mean performance in all levels of uncertainty and 1. Figure 20 displays the deterioration of the solutions for the uncertainty levels simulated. It shows that the level of deterioration of the aggregate performance of the rules decreases as the uncertainty increases. Furthermore, this figure demonstrates that the loss of quality of setup-oriented methods (around 15%) is significantly lower than that of other techniques (around 40%). Finally, at a desegregated level, Figure 20 shows that the performance of the results of the best performing evaluated method (*MMS*) can achieve equal or better results than the decentralized algorithm for high uncertainty levels (cv = 0.8 or cv = 1.0).

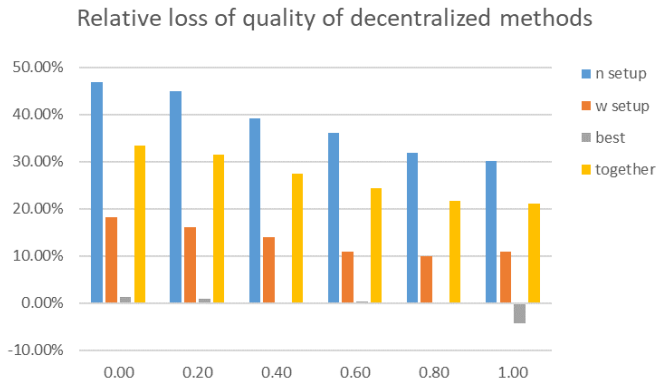

Figure 20: Loss of quality of scheduling methods in real-world instances and in the research of L&S

To summarize, there are clear differences between the two simulated sets of problems caused mainly by the introduction of sequence-dependent setup times. In fact, it is verified that the introduction of setup times dilutes the processing times' uncertainty, diminishing the divergence of performance of static and dynamic sequences. Furthermore, the results show that the best performing scheduling methods are setup-oriented, revealing a significant advantage over the techniques studied by L&S. Finally, the loss of quality of scheduling solutions provided by agile and decentralized methods seems fairly large if all the studied techniques are included. However, it reduces as the uncertainty level increases. It is also noteworthy that a more desegregated view, including only the setup-oriented methods, shows a reasonable degree of solution's deterioration. Focusing only on the best decentralized performing method, it can be concluded that it achieves comparable or superior results than the centralized and complex algorithm.

These observations strengthen the initial hypothesis that setup times are a critical factor in scheduling for problems with sequence-dependent setup times. In the second experiments' set, the influence of setup times is deeply explored.

5.2 Second experiments' test

The instances of the real-world case and the set of 53 instances studied by L&S have intrinsic differences, which are clearly reflected on the performance of the evaluated scheduling methods. As stated in the previous section, setup-oriented methods techniques obtained significantly better performances than the others for the real-world instances. Thereby, it is important to understand the impact of the setup times in the selection of the best method, since sequence-dependent setup times are the most relevant differences between the problems. It is equally important to understand whether this discrepancy of performance is only caused by the introduction of sequence-dependent setup times, or there are other factors that should be taken into account.

Therefore, sequence-dependent setup times were generated for the instances investigated by L&S in order to study the comparable problems with the real-world instances. Four levels of setup times were created to better perceive their effect over the performance of the methods. It is noteworthy that since the effect of the static and dynamic schedules was a topic already discussed in the previous section, this part only focuses on dynamic approaches to evaluate other research questions.

5.2.1 Results

The experiments for each setup level were simulated in the exact same conditions as previous experiments' set. However, previously the results were displayed as the fraction value of the makespan and the optimal or quasi-optimal solution for the respective problem, and now those values are not known. Thereby, the lower bound of each problem for each setup level was computed using a simple method in which the lower bound is the maximum machine makespan of the problem, considering that each operation is in the machine the processing time plus the minimum possible setup time. It is noteworthy that to keep the coherency and do a fair comparison of the results, the same lower bounds were computed for the real-world instances.

The average of the mean performance obtained at each uncertainty level for all setup levels and for the real-world instances is summarized in Table 8.

Table 8: Mean scheduling methods' performance of all levels of uncertainty methods for each setup level

Method	SL = 0,25	SL = 0,5	SL = 0,75	SL = 1	Real-world Case
FCFS	1,565	1,593	1,619	2,233	1,781
SPT	1,589	1,629	1,661	2,274	1,869
LPT	1,709	1,729	1,740	2,357	1,841
LSD	1,557	1,598	1,630	2,215	1,765
MWF	1,467	1,500	1,523	2,073	1,768
LTR	1,513	1,543	1,568	2,131	1,835
MOR	1,512	1,540	1,566	2,160	1,745
SSPT	1,577	1,617	1,648	2,235	1,692
MMS	1,492	1,516	1,537	2,098	1,306
SNSPT	1,586	1,625	1,654	2,233	1,467
GA	1,578	1,621	1,572	2,227	1,782
CH	1,599	1,621	1,629	2,213	1,353
GH	1,600	1,622	1,630	2,217	1,342

Figure 21 puts together the four uncertainty levels to display the mean performance of each method for a given level processing times' uncertainty.

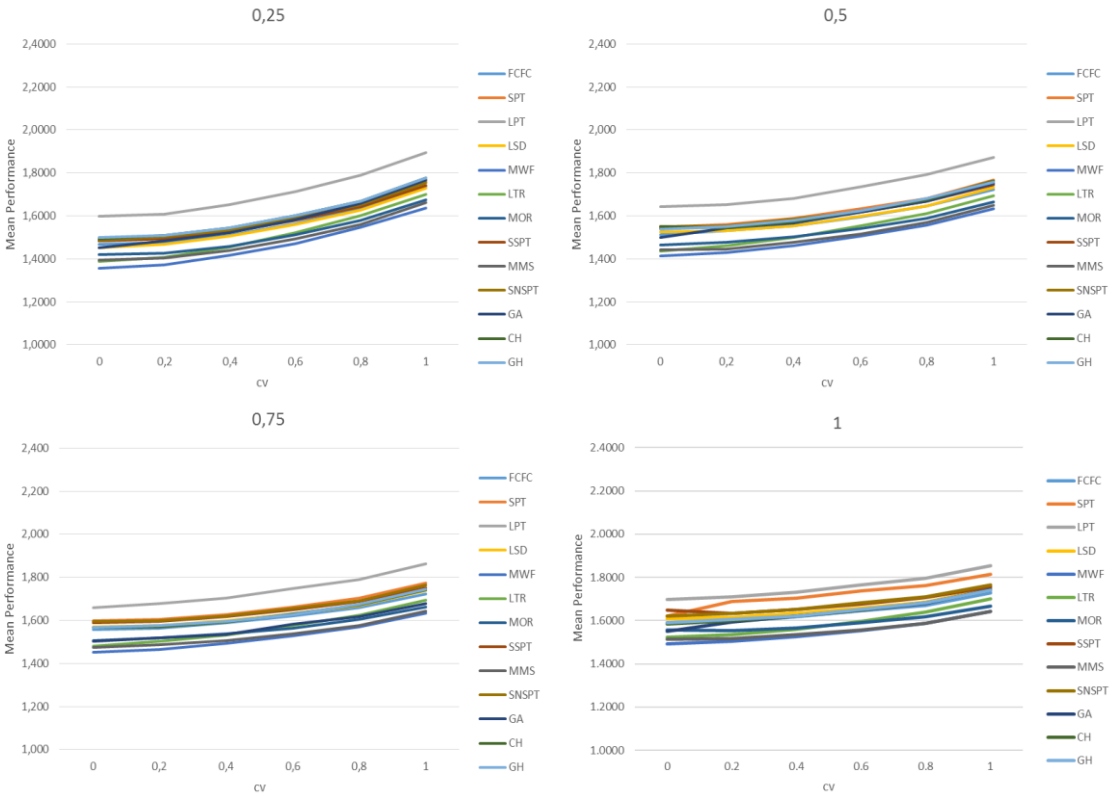

Figure 21: Mean performance of each scheduling method for each setup level

5.2.2 Discussion of results

The discussion of the results from the second experiments' test focuses on the investigation of the influence of sequence-dependent setup times in the performance of the scheduling methods.

Analysing the global performance of the rules through the observation of Figure 21, it is verified that for a given setup level the slope of the curves reduces as the level of setup increases. This indicates that the effect of the processing times' uncertainty is less significant for higher levels of setups due to the loss of relative importance of processing time over the final makespan.

Focusing on the relative performance of the rules, the results indicate that there are not relevant differences from each setup level, and the relative performance of each method is very stable.

One interesting observation is that *GH* and *CH*, which are rules that only focus on setup times, seem to improve gradually for higher levels of setups. Despite in the real-world case these methods showed significantly better performances over the no setup time oriented method, the same conclusions cannot be taken for these experiments. Thereby, sequence-dependent setup times seem to be an important factor for the selection of the scheduling method but it is not as critical as expected.

Effectively, the results of the second experiment show that *MWF* is always the best performance rule in all levels of setup, *MMS*, *LTR* and *MOR* being the following best performing techniques. It is noteworthy that all these methods consider the future work of the jobs, giving priority to those which have either more time to be processed or more machines

to visit. This analysis suggests that the work in following machines is important to the selection of the methods.

In contrast, *SPT* and *LPT* have the worst performances for almost all setup levels. Also, other processing time oriented methods, such as *SSPT* and *SNSPT* do not have very good relative performances. It is also noteworthy that *LSD* is the method which reduces the most its relative ranking. These observations indicate that processing time is not a good parameter to focus on the selection of the scheduling technique. Figure 22 displays the relative ranking of mean performance of the studied scheduling methods.

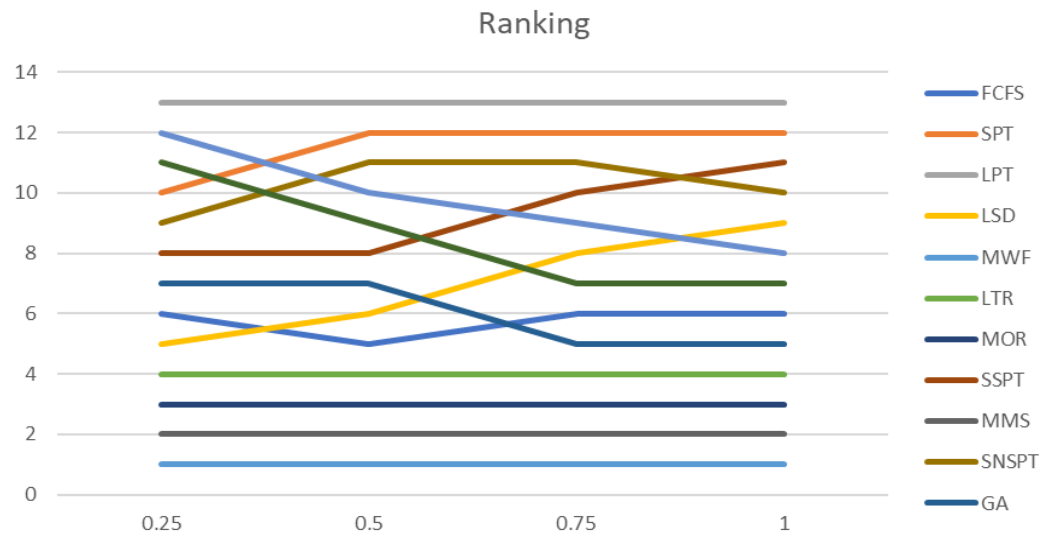

Figure 22: Relative ranking of scheduling method for each setup level

In fact, the conclusions taken in the of the real-word instances is quite different, since the performance of setup-oriented methods was considerably better than the others which do not occurs in this experiment.

Nevertheless, it is evident that setup times influences the selection of the method since, the higher the setup times, the more important setups are, even though in these set of experiments, setup oriented methods do not show very good performances, mainly for lower setup levels. This indicates that there may be other attributes that differentiate the problems and that are as critical or possibly more critical for the selection of the best scheduling method to apply. Comparing the performance of no setup-oriented methods, *MWF* and *MOR* are also the best performing methods for the real-world instances, which strengthen the idea that future work of jobs should be considered on the selection of the method. Another fact that supports this idea is the performance of *MMS*, which is one of the best scheduling techniques for both the instances studied by L&S and the real-world case. This rule derives from both setup and following work of job.

In fact, the instances studied by L&S and the real-world problems have clear differences which can justify the difference of quality of setup-oriented methods. Concerning the number of orders, the real-world instances have a considerably higher number of orders per machine which makes that each queue have a higher number of orders to schedule at a given time. Since the methods are applied locally, having a higher number of orders, setup-oriented methods have a large probability to find good solutions in terms of the queue total setup time. Thus, in problems with high number of orders per machine setup-oriented methods are more likely to obtain good solutions.

Regarding the loss of quality of the scheduling methods, it is noticeable that for higher setup times the performance of the methods deteriorates considerably. In fact, as shown in Figure 23 quality decreases gradually with as setup times increase. It is also possible to

visualize that setup-oriented methods have a higher loss of quality for the lower setup levels (0.25 and 0.5), but lower for higher setup levels (0.75 and 1) than no setup-oriented methods. This indicates that for a high portion of setup relatively to processing time, decentralized methods should be investigated in more detail, when compared to centralized ones.

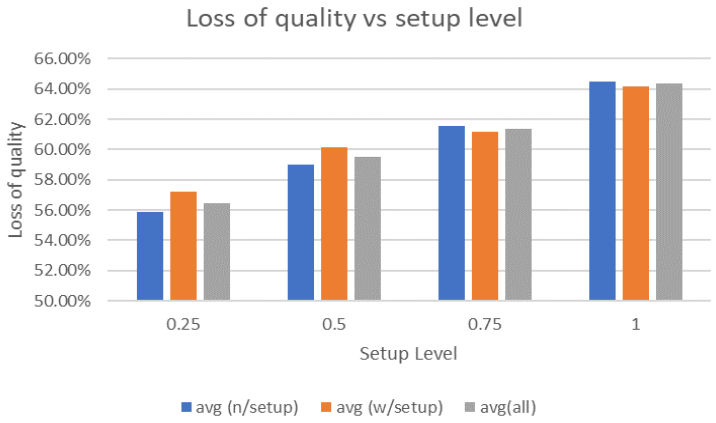

Figure 23: Loss of quality relatively to the most optimist lower bound at each setup level

In order to have another indicator about the performance of each scheduling method, another optimality criteria were employed: the percentage of total working time of the machines. As it was mentioned in section 3.3.2, makespan can be myopic, which means that optimizing one machine does not assure that the remaining machines are optimized. Thereby, by computing the percentage of time that all the machines are actually working, it is possible to have perspective information about whether the methods are being efficient in certain time period or not. This measure is especially critical in a real word case, where the schedulers should not provide a one-shot schedule with an excellent performance, but that disregards a right balance between the utilization of machines. Instead, it is necessary to generate a schedule with good performance and properly balanced, so it can accommodate future orders which will arrive to the shop-floor. The results obtained with this metric as well as the relative ranking of the methods for each level of setup are displayed in Table 9.

Table 9: Percentage of working time and relative ranking of each method for the four setup levels investigated

Method	SL = 0,25		SL = 0,5		SL = 0,75		SL = 1	
FCFC	0,7849	8	0,7353	10	0,7020	7	0,6769	3
SPT	0,7947	3	0,7390	3	0,7014	10	0,6733	11
LPT	0,7700	13	0,7270	13	0,6961	13	0,6720	13
LSD	0,7941	4	0,7392	1	0,7020	8	0,6741	8
MWF	0,7834	11	0,7333	11	0,6996	12	0,6737	10
LTR	0,7802	12	0,7325	12	0,6998	11	0,6752	7
MOR	0,7849	9	0,7366	8	0,7021	6	0,6766	5
SSPT	0,7952	2	0,7388	4	0,7019	9	0,6729	12
MMS	0,7867	6	0,7371	7	0,7039	1	0,6778	1
SNSPT	0,7953	1	0,7391	2	0,7032	4	0,6739	9
GA	0,7923	5	0,7378	6	0,7023	5	0,6759	6
CH	0,7851	7	0,7364	9	0,7034	3	0,6774	2
GH	0,7844	10	0,7381	5	0,7036	2	0,6767	4

An analysis of this table enables to perceive that the bigger the setup level, the worse is the performance the schedules for this optimality criteria. In fact, when setup times' relative

proportion to processing times increases, machines spend more time being prepared than they are processing in a relative scale.

Another important observation is that setup-oriented methods, such as *CH, GH* and MMS increase their relative performance, since they prioritize orders with less setup and thus machines will try to minimize the times of preparation to being actually producing. In contrast, processing times' oriented methods decrease their performance as setup level increases. In fact, giving preference to shorter orders, makes that more orders are processed in a given time period and more setups will be done. Thereby, the total setup time of the system is increased and the percentage of working time reduces.

It is also interesting to observe that M*WF*, that performed reasonably well in both the real-wold instances and the setup of problems of L&S, has a poor performance regarding this metric. Contrarily, MMS which is the best performing rule for real-world instances and one of the bests in the second experiments' tests, can perform quite well in this criterion. In fact, its relative performance increases as the setup time increase, since it considers the setup time.

To summarize, in the previous experiments setup-oriented-methods showed significantly better performances over other techniques, but a deeper analysis of the impact of setup times was necessary. In this second set of experiments, four levels of setups were evaluated and the results evidenced that, effectively, setup times affect the performance of the scheduling methods. It was verified that that relative ranking of setup-oriented methods, such as *CH* and *GH*, tend to increase as the setup level increases. Furthermore, it was observed that the quality of the scheduling solutions reduces considerably at a certain level of setup, which indicates that setup times should be wisely studied when opting for decentralized instead of centralized methods.

However, the implications of setup times are not so large as expected. In fact, for the same proportion of setup and processing time the setup-oriented methods did not evidence a great advantage over the other techniques as they did in the first experiments of the real-world case. Actually, the analysis of the results showed that there are other factors, such as the following work of jobs, that may be as critical as setup times for the performance of the methods, since rules like *MOR* and *MWF* showed good performances for both the real-word instances and the instances of L&S. This idea is further supported by the fact that one of the best performing rules for both experiments was *MMS*, which considers not only the setup time, but also the number of machines remaining of given a job.

Concerning the optimality criteria of production capacity, results enabled to understand that scheduling rules, that have great performances in makespan, may not guarantee the best balance between the machines. This is evidenced by the performance of rules like *MWF*, which showed good results in terms of makespan, but does not assure a sustainable schedule in a continuous production horizon. This can have serious implications in a real case schedule, which must be prepared to accommodate new arriving orders.

6 Evolved Methods

The investigation performed in the first experiments' set evidenced that the different characteristics of the problems can deeply influence the performance of scheduling methods. More specifically, setup-oriented methods showed significantly better performances over the other techniques when applied to real-world problems with sequence-dependent setup times.

The study of the second experiments' set aimed at understanding the influence of setup times in the performance of scheduling methods applied to the problems of the research of L&S, with generated sequence-dependent setup times. However, in contrast to what was initially expected, the performance of sequence-dependent methods was not as good as in the real-world case instances. In fact, an improvement of performance of these methods as setup times increase was verified, but other techniques that perform well in L&S research, maintain their good performances even with high levels of setup time. Therefore, besides this parameter, other factors are believed to influence the selection of the best scheduling method, depending on the specific characteristics of the problem.

In order to develop a rule which considers several characteristics and can easily adapt to the applied environment, a new adaptive method was evolved using genetic programming. In this section, the design of the method is presented as well as the obtained rule. Next, the method is applied to the 53 instances of L&S and the results are compared with those obtained in the previous experiments.

6.1 Method design

The development of the genetic programming model was done using Heuristic Lab, which is a software that provides a framework to develop heuristic and evolutionary algorithms for a wide range of applications. In this research, the purpose is to develop a simulation optimization technique that uses simulation as an evaluation function of the quality of the methods generated by the evolutionary algorithm. Thus, the genetic programming model developed in Heuristic lab, which runs the optimization algorithm, was connected to Anylogic, where the solutions scheduling methods created were assessed.

The genetic programming model in heuristic lab is composed by a genetic algorithm, in which solutions are encoded by symbolic tree, which represents mathematical expressions. Fundamentally, these expressions represent functions used to evaluate each job in the queue, associating a priority index to each member. Thereby, a new queue is generated by sorting the jobs in queue in the descending order of its priority indexes. It is also important to refer that each generated method is applied dynamically, which means that every time a new job arrives to the queue, the queue is revaluated by the scheduling method.

The objective of this genetic programming model is to evolve a simple and clear agile method, that adapts to various problems characteristics, and thus only simple operations were included. Consequently, the symbolic tree only involves the four main arithmetic functions:

addition, subtraction, multiplication, and division; The terminals can only include four variables: current processing time(Pt), number of operations remaining(NOps), remaining processing time(RPt) and setup time(St). In the selection of these variables the goal was to include the ones enabling the genetic programming model to generate any rule that was investigated by the two experiments 'set. In truth, the chosen four variables only do not allow the model to generate *SNSPT* and *LTR* which considers other operators as the average. Regarding the size of the symbolic tree, the algorithm limits the generation of solutions to tree with maximum depth 5 and maximum length 32.

Since the simulation is relatively computational heavy, a small group of instances was selected to be used in the genetic programming. This selection aimed at representing the whole range of problems and thus it includes either instances with large number of jobs, or large number of machines or even a large proportion of jobs to machine. Thereby, three instances of L&S and of the real-wold case were selected. Furthermore, to represent effect of stochasticity in processing times, the medium level of uncertainty (cv = 0.6) was introduced, simulating 15 perturbed problems for each instance. To include the influence of setup times on the performance of the methods, each instance of L&S was simulated considering both the highest level of setup times (SL=1) and the lowest level (SL=0,25). At the end, for each instance of L&S 30 problems are evaluated and 15 for the instances of the real-world case, which results in 120 problems are assessed.

The parametrization of the genetic algorithm was done performing small sets experiments with a reduced number of instances, where some modifications of the parameters are tested. Also, the parameterization took into consideration the orientation provided by the paper Tsai and Fu (2014). The final algorithm uses a population size of 30, with 2 elite members and performs 12 generations. It considers a crossover probability of 90% and a mutation probability of 15 %. This parameterization results in 43200 simulation's runs necessary.

6.2 Rule evolved

The genetic programming model generated a rule that only includes 3 of the 4 possible variables, which demonstrates that processing times seem not to be relevant to improve the performance of the methods. The rule created is displayed in Figure 24.

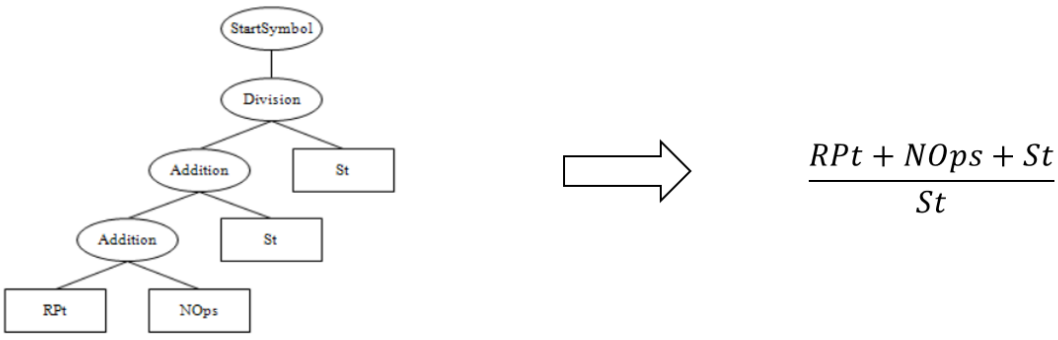

Figure 24: Evolved rule using Genetic Programming

The Figure 25 displays the performance of the genetic algorithm, and it possible to visualize the convergence of the average quality of the solutions to the best-found solution. In fact, the average quality of the rules quickly approximates of the best-found, since from the generation 7, the average quality is lower than 1.5.

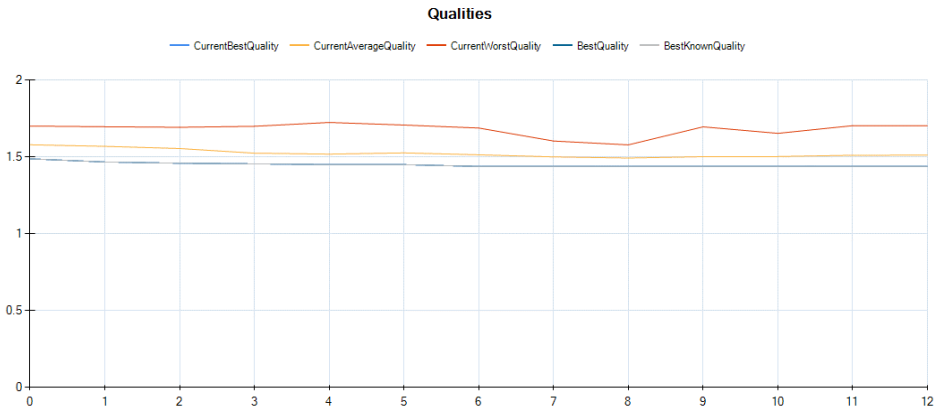

Figure 25: Quality of the solutions evaluated in the Genetic Programming model

6.3 Tests and discussion of results

The evolved rule was applied in all the 53 instances for both the highest setup level and the lowest setup level and for the 5 instances of the real-world case. The obtained results for some of the best performing rules and the worst in each group of problems are displayed in Figure 26.

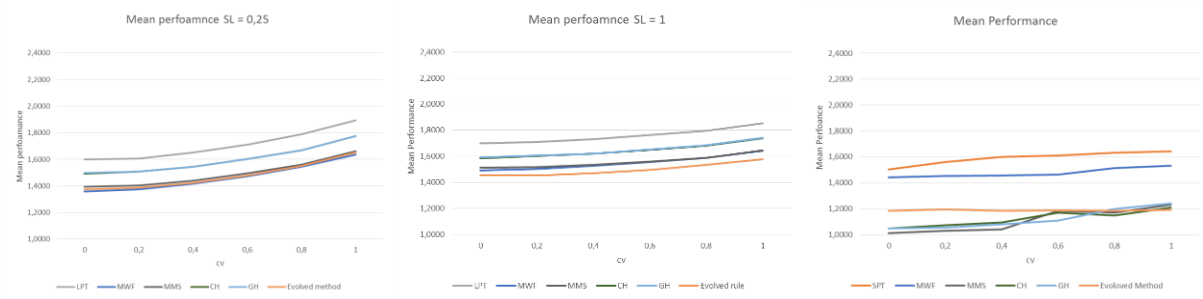

Figure 26: Mean performance of the scheduling method for the benchmark instances with setup level of 0,25(left), 0,5(center) and for the instances of the real-world case (right)

Analysing Figure 26 it is possible to understand that the evolved rule achieved generally good results in terms of makespan performance for each one of the three groups of problems where it was applied. More precisely, for the benchmark instances with setup level of 0,25, it is the second best performing rule for all levels of uncertainty. In the case of setup level of 1, the evolved method achieved the best performance with a significant difference from the other methods. Finally, concerning the real-world instances, the rule demonstrates to be very robust to processing times' uncertainty, since its performance does not deteriorate significantly as the uncertainty increases. In fact, for the lower levels of uncertainty, the evolved method performs well, but it cannot achieve the results obtained by *MMS*, *CH* or *GH*. However, as uncertainty increases, the performance of the evolved method converges to the performance of these methods.

To conclude, the rule generated using the genetic programming model achieves high performances even in presence of both sequence-dependent setup-times and processing times' uncertainty.

7 Conclusions and future work

The new demand requirements forced by the global economy, and the increasing complexity of productions systems due to the vision of the fourth Industrial revolution impose new requirements for manufacturing. Traditional production process modelling techniques rely heavily on central decision-making structures, which present numerous disadvantages when they must deal with real systems, whose agility and responsiveness are fundamental to manage all kind of disturbances. Decentralization of decision-making may be an interesting solution to overcome these issues.

In light of the research question exposed in section 1.2, this dissertation extends the work of L&S, who evaluated centralized and decentralized scheduling methods for a range of benchmark instances, considering processing time variation. In this project, the same methodology was applied to real-world instances to verify whether the conclusions of L&S remain valid for more complex problems, namely with introduction of sequence-dependent setup times. For that propose an adaptive agent-based simulation model was developed to evaluate the methods proposed by L&S. Additionally, more advanced setup-oriented methods were implemented and tested in problems with different characteristics of jobs and machines. Two sets of experiments were performed. In the first the scheduling methods were applied to static and dynamically to the complex instances of a real-world case. In the second, the impact of sequence-dependent setup times in the performance of the agile scheduling methods was investigated, by generating four levels of setups for the benchmark instances studied by L&S. This enabled to perceive the evolution of the scheduling methods' performance with the gradual increment of setup times, establishing a parallel comparison with the results in the real-world instances. Finally, a genetic programming model was implemented to develop an agile and responsive scheduling methods which aims at weighting the most relevant attributes according to the intrinsic characteristics of the problem where it is applied.,

The first research question "Which cases motivate a decentralized decision-making structure?" has been addressed comprehensively by assessing the advantages and disadvantages of centralized and decentralized decision structures and making a survey of the best fit scenarios and applications of decentralized systems. It is observed that highest profit of decentralize decision-making structures is achieved when applied to highly complex manufacturing systems, where large amounts of data are generated, such as the semiconductor industry.

The second research question "What is the quality loss of decentralized scheduling methods when compared to centralized solutions?" has been treated in the first set of experiments, where the scheduling methods were applied to the real-world instances. The results showed that as uncertainty increases, the loss of quality of decentralized methods diminishes. It was also concluded that despite the average loss of quality of decentralized methods was fairly large (around 25 %), a more desegregated analysis revealed that setup-oriented methods had an acceptable loss of quality (around 15%). The results were even more satisfactory when analysing the best performing method, which exhibited better performances

than the complex and centralized technique for high uncertainty levels, and comparable performances at the lower levels.

The third research question is about the advantage of performance of dynamic over static scheduling methods and it was tackled in first set of experiments. In fact, in more complex and larger instances, with sequence-dependent setup times, the advantage of using dynamic methods was lower than the results provided by L&S, even in high levels of processing times' uncertainty. This is related to the fact that setup times reduce the importance of processing times as an influence factor of the makespan and thus, their variations are not so explicitly revealed in the final performance.

Regarding the fourth research question, about the implications of sequence-dependent setup times in the quality of the solutions of decentralized and agile scheduling methods, the first set of experiments enabled to verify that setup-oriented methods have significantly better performances over the methods investigated in the paper of L&S. In fact, *MMS* is the best performing method for almost all uncertainty levels. Concerning the methods investigated by L&S, MOR and MWF showed good relative results, as it concluded in the analysis of L&S. The second experiment's set enabled to perceive that the performance of setup-oriented methods improves with the level of setup, even though these methods did not show results as good as when applied to the real-world instances. Effectively, rules who perform well without setups, such as *MOR* and *MWF* and *MMS* were the ones who perform the best.

The fifth research question "Are sequence dependent setup times a critical factor when optimizing for decentralize methods?" has been addressed in the second set of experiments. The observation of the results showed that the performance of setup-oriented methods improves as the setup level increases, which indicates that setup times influence the performance of the scheduling methods. However, even for the high setup levels, the best performing rules still were rules which consider other attributes besides setup times, such as the number of operations remaining or the processing time reaming. This indicates that other characteristics of the problem, besides setup times, are also critical to the selection of the best scheduling method. Regarding the loss of quality of decentralized and agile methods, the second set of experiments showed that as setup level increases, the performance of decentralized and agile methods deteriorate relatively to the lower bound solutions of the problems, which means that in presence of high level of sequence-dependent setups relatively to processing times, the choice for decentralized methods needs to be evaluated carefully.

The last research question, about the best performing methods in presence of sequence-dependent setup times, has been addressed in both the first and the second sets of experiments. In fact, in the first set of experiments *MMS* has shown to be the best performing rule, but other setup-oriented methods, such as *CH* and *GH*, also showed good results. In the second experiments set, rules which consider the future work of the jobs are the ones which perform the best. More precisely, *MWF*, *MOR* and *MMS* were the best performing methods.

Finally and despite the heavy computational effort, the evolved method using the genetic programming technique showed not only a good behavior in presence of large sequence-dependent setup times, but also a good robustness against processing times' uncertainty.

In the scope of this dissertation, the influence sequence-dependent setup times was assessed as the critical factor to impact the performance of scheduling methods in complex manufacturing environments. In a broader research, it would be relevant to explore other characteristics of the problems to assess their implications on the performance of the scheduling methods. Moreover, other agile decentralized methods should be evolved to achieve comparable performances to those of centralized methods, even at low uncertainty levels, motivating the choice of decentralization.

References

Adams, J., Balas, E. and Zawack, D. (1988) 'The Shifting Bottleneck Procedure for Job Shop Scheduling'. Management Science, 34(3).

Aelker, J., Bauernhansl, T. and Ehm, H. (2013) 'Managing complexity in supply chains: A discussion of current approaches on the example of the semiconductor industry', *Procedia CIRP*. Elsevier B.V., 7, pp. 79–84. doi: 10.1016/j.procir.2013.05.014.

Anylogic Web Page (2017). Available at: https://www.anylogic.com/use-of-simulation/agent-based-modeling/ (Accessed: 25 June 2017).

Applegate, D. and Cook, W. (1991) 'A computational study of the job-shop scheduling problem', *ORSA Jounal on Computing*, 3(August 2015), pp. 149–156. doi: 10.1287/ijoc.3.2.149.

Arzi, Y. and Raviv, D. (1998) 'Dispatching in a workstation belonging to a re-entrant production line under sequence-dependent set-up times', *Production Planning & Control*, 9(7), pp. 690–699. doi: 10.1080/095372898233696.

Aytug, H., Lawley, M. A., McKay, K., Mohan, S. and Uzsoy, R. (2005) 'Executing production schedules in the face of uncertainties: A review and some future directions', *European Journal of Operational Research*, 161(1), pp. 86–110. doi: 10.1016/j.ejor.2003.08.027.

Barzegar, B., Motameni, H. and Bozorgi, H. (2012) 'Solving flexible job-shop scheduling problem using gravitational search algorithm and colored Petri net', *Journal of Applied Mathematics*, 2012. doi: 10.1155/2012/651310.

Bean, J. C., Birge, J. R., Mittenthal, J. and Noon, C. E. (1991) 'Matchup Scheduling with Multiple Resources, Release Dates and Disruptions', *Operations Research*, 39(3), pp. 470–483. doi: 10.1287/opre.39.3.470.

Blackstone, J. H., Phillips, D. T. and Hogg, G. L. (1982) 'A state-of-the-art survey of dispatching rules for manufacturing job shop operations', *International Journal of Production Research*, 20(1), pp. 27–45. doi: 10.1080/00207548208947745.

Blum, C. and Roli, A. (2003) 'Metaheuristics in combinatorial optimization: overview and conceptual comparison', *ACM Computing Surveys*, 35(3), pp. 189–213. doi: 10.1007/s10479-005-3971-7.

Bongaerts, L. (1998) *Integration of Scheduling and Control, 24 European Symposium on Computer Aided Process Engineering*. doi: 10.1016/B978-0-444-63456-6.50072-7.

Braun, P. (2014) 'Analysis of production planning and control systems for automotive powertrain assembly lines concerning the level of decentralization in the context of Industrie 4 . 0'.

Brettel, M., Friederichsen, N. and Keller, M. (2014) 'How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective', *International Journal of*, 8(1), pp. 37–44. doi: scholar.waset.org/1999.8/9997144.

Chryssolouris, G. and Subramaniam, V. (2001) 'Dynamic scheduling of manufacturing job shops using genetic algorithms', *Journal of Intelligent Manufacturing*, 12, pp. 281–293.

Church, L. K. and Uzsoy, R. (1992) 'Analysis of periodic and event-driven rescheduling policies in dynamic shops', *International Journal of Computer Integrated Manufacturing*, 5(3), pp. 153–163. doi: 10.1080/09511929208944524.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2001) *Introduction to Algorithms*. MIT Press & McGraw–Hill.

Dilts, D. M., Boyd, N. P. and Whorms, H. H. (1991) 'The evolution of control architectures for automated manufacturing systems', *Journal of Manufacturing Systems*, 10(1), pp. 79–93. doi: 10.1016/0278-6125(91)90049-8.

Van Dyke Parunak, H. (1991) 'Characterizing the manufacturing scheduling problem', *Journal of Manufacturing Systems*, 10(3), pp. 241–259. doi: 10.1016/0278-6125(91)90037-3.

Figueira, G. and Almada-Lobo, B. (2014) 'Hybrid simulation-optimization methods: A taxonomy and discussion', *Simulation Modelling Practice and Theory*. Elsevier B.V., 46, pp. 118–134. doi: 10.1016/j.simpat.2014.03.007.

Floudas, C. A. and Lin, X. (2005) 'Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications', *Annals of Operations Research*, 139(1), pp. 131–162. doi: 10.1007/s10479-005-3446-x.

Fohler, G. and Fohler, G. (2015) 'What is the Difference Between Offline and Online Scheduling ?', (March), pp. 1–3.

Geiger, C. D., Uzsoy, R. and Aytuğ, H. (2006) 'Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach', *Journal of Scheduling*, 9(1), pp. 7–34. doi: 10.1007/s10951-006-5591-8.

Gendreau, Michel and Potvin, J.-Y. (2010) *Handbook of Metaheurisitcs*. Springer. doi: 10.1007/978-1-4614-1900-6.

Goldberg, D. E., Korb, B. and Deb, K. (1989) 'M essy Genetic Algorithms : Motivation , Analysis , and First Results', *Engineering*, 3, pp. 493–530.

Golmakani, H. R. and Birjandi, A. R. (2013) 'A two-phase algorithm for multiple-route job shop scheduling problem subject to makespan', *International Journal of Advanced Manufacturing Technology*, 67(1–4), pp. 203–216. doi: 10.1007/s00170-013-4767-6.

Gonçalves, J. F. and Resende, M. G. C. (2011) 'Biased random-key genetic algorithms for combinatorial optimization', *Journal of Heuristics*, 17(5), pp. 487–525. doi: 10.1007/s10732-010-9143-1.

Graham, R. L., Lawler, E. L., Lenstra, J. K. and Kan, A. H. G. R. (1979) 'Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey', *Annals of Discrete Mathematics*, 5(C), pp. 287–326. doi: 10.1016/S0167-5060(08)70356-X.

Gromicho, J. A. S., Van Hoorn, J. J., Saldanha-Da-Gama, F. and Timmer, G. T. (2012) 'Solving the job-shop scheduling problem optimally by dynamic programming', *Computers and Operations Research*. Elsevier, 39(12), pp. 2968–2977. doi: 10.1016/j.cor.2012.02.024.

Haupt, R. (1989) 'A survey of priority rule-based scheduling', *OR Spektrum*, 11(1), pp. 3–16. doi: 10.1007/BF01721162.

Held, M. and Karp, R. M. (1962) 'A Dynamic Programming Approach to Sequencing Problems', *Journal of the Society for Industrial and Applied Mathematics*, pp. 196–210. doi: 10.1137/0110015.

Hermann, M., Pentek, T. and Otto, B. (2016) 'Design principles for industrie 4.0 scenarios', *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2016–March, pp. 3928–3937. doi: 10.1109/HICSS.2016.488.

Hildebrandt, T., Heger, J. and Scholz-Reiter, B. (2010) 'Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach', *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 257–264. doi: doi:10.1145/1830483.1830530.

Hilier, F. and Lieberman, G. (2015) *Introduction to Operational Research, Introduction to Operational Research*. doi: 10.2307/2077150.

Ho, N. B. and Tay, J. C. (2005) 'Evolving dispatching rules for solving the flexible job-shop problem', *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 3, pp. 2848–2855. doi: 10.1109/CEC.2005.1555052.

Hulsmann, M. and Windt, K. (2007) *Understanding Autonomous Cooperation and Control in Logistics*. Springer.

Jacoboni, C. and Lugli, P. (2012) *The Monte Carlo Method for Semiconductor Device Simulation*. Springer Science & Business Media.

Johnson, S. M. (1954) 'Optimal two- and three-stage production schedules with setup times included', *Naval Research Logistics Quarterly*, 1(1), pp. 61–68. doi: 10.1002/nav.3800010110.

Kaban, A. K., Othman, Z. and Rohmah, D. S. (2012) 'Comparison of dispatching rules in job-

shop Schedulingproblem Usingsimulation: A case study', *International Journal of Simulation Modelling*, 11(3), pp. 129–140. doi: 10.2507/IJSIMM11(3)2.201.

Khayat, G. El, Langevin, A. and Riopel, D. (2006) 'Integrated production and material handling scheduling using mathematical programming and constraint programming', *European Journal of Operational Research*, 175(3), pp. 1818–1832. doi: 10.1016/j.ejor.2005.02.077.

Kim, M. H. and Kim, Y.-D. (1994) 'Simulation-based real-time scheduling in a flexible manufacturing system', *Journal of Manufacturing Systems*, 13(2), pp. 85–93. doi: http://dx.doi.org/10.1016/0278-6125(94)90024-8.

Kochhar, S. and Morris, R. J. T. (1987) 'Heuristic methods for flexible flow line scheduling', *Journal of Manufacturing Systems*, 6(4), pp. 299–314. doi: 10.1016/0278-6125(87)90006-9.

Koulamas, C. (1998) 'A new constructive heuristic for the flowshop scheduling problem', *European Journal of Operational Research*, 105(1), pp. 66–71. doi: 10.1016/S0377-2217(97)00027-1.

Koza, J. R. (1992) *Genetic programming: On the programming of computers by means of natural selection*, *Biosystems*. The MIT Press. doi: 10.1016/0303-2647(94)90062-0.

Lawrence, S. (1984) *Resouce constrained project scheduling: an experimental investigation of heuristic scheduling techniques*.

Lawrence, S. R. and Sewell, E. C. (1997) 'Heuristic, optimal, static, and dynamic schedules when processing times are uncertain', *Journal of Operations Management*, 15(1), pp. 71–82. doi: 10.1016/S0272-6963(96)00090-3.

Leitão, P. (2009) 'Agent-based distributed manufacturing control: A state-of-the-art survey', *Engineering Applications of Artificial Intelligence*, 22(7), pp. 979–991. doi: 10.1016/j.engappai.2008.09.005.

Leung, J. Y.-T. (2004) *Handbook of Scheduling, Algorithms, Models and Performance Analysis, Methods*.

Liao, C.-J. and You, C.-T. (1996) 'Operational Research Society is collaborating with JSTOR to digitize, preserve, and extend access to Journal of the Operational Research Society. ® www.jstor.org', *Journal of the Operational Research Society*, 47, pp. 697–701.

Lou, P., Liu, Q., Zhou, Z., Wang, H. and Sun, S. X. (2012) 'Multi-agent-based proactive-reactive scheduling for a job shop', *International Journal of Advanced Manufacturing Technology*, 59(1–4), pp. 311–324. doi: 10.1007/s00170-011-3482-4.

Manne, A. S. (1960) 'On the Job-Shop Scheduling Problem', *Operations Research*, 8(2), pp. 219–223.

Mehta, S. V. and Uzsoy, R. M. (1998) 'Predictable scheduling of a job shop subject to breakdowns', *IEEE Transactions on Robotics and Automation*, 14(3), pp. 365–378. doi: 10.1109/70.678447.

Miyashita, K. (2000) 'Job-Shop Scheduling with Genetic Programming', in *GECCO'00 Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*. Las Vegas, Nevada: Morgan Kaufmann Publishers Inc. San Francisco, CA, USA ©2000, pp. 505–512.

Muth, J. F. and Thompson, G. L. (1963) *Industrial Scheduling*. Englewood Cliffs, N.J. : Prentice-Hall.

Nakasuka, S. and Yoshida, T. (1992) 'Dynamic scheduling system utilizing machine learning as a knowledge acquisition tool', *International Journal of Production Research*, 30(2), pp. 411–431. doi: 10.1080/00207549208942903.

Nguyen, S., Mei, Y. and Zhang, M. (2017) 'Genetic programming for production scheduling: a survey with a unified framework', *Complex & Intelligent Systems*. Springer Berlin Heidelberg, 3(1), pp. 41–66. doi: 10.1007/s40747-017-0036-x.

Nguyen, S., Zhang, M., Member, S., Johnston, M., Tan, K. C. and Member, S. (2012) 'A Computational Study of Representations in Genetic Programming to Evolve Dispatching Rules for the Job Shop Scheduling ... A Computational Study of Representations in Genetic Programming to Evolve Dispatching Rules for the Job Shop Scheduling Problem', 17(January), pp. 621–639. doi: 10.1109/TEVC.2012.2227326.

Ouelhadj, D. and Petrovic, S. (2009) 'A survey of dynamic scheduling in manufacturing systems', *Journal of Scheduling*, 12(4), pp. 417–431. doi: 10.1007/s10951-008-0090-8.

Ovacik, I. M. and Uzsoy, R. (1997) *Decomposition methods for complex factory scheduling problems*.

Panwalkar, S. S. and Iskander, W. (1977) 'A Survey of Scheduling Rules', *Operations Research*, 25(1), pp. 45–61. doi: 10.1287/opre.25.1.45.

Pickardt, C. W. and Branke, J. (2012) 'Setup-oriented dispatching rules – a survey', *International Journal of Production Research*, 50(20), pp. 5823–5842. doi: 10.1080/00207543.2011.629634.

Pickardt, C. W., Hildebrandt, T., Branke, J., Heger, J. and Scholz-Reiter, B. (2013) 'Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems', *International Journal of Production Economics*, 145(1), pp. 67–77. doi: 10.1016/j.ijpe.2012.10.016.

Pinedo, M. L. (2008) *Scheduling*. doi: 10.1007/978-0-387-78935-4.

Resende, M. G. C. and Gonçalves, J. F. (2013) 'A Biased Random-key Genetic Algorithm for Job-Shop Scheduling', (February 2011), pp. 1–25.

Sauter, R., Bode, M. and Kittelberger, D. (2015) '"How Industry 4.0 Is Changing How We Manage Value Creation"', *Horváth & Partners*. Available at: https://www.horvath-partners.com/fileadmin/horvath-partners.com/assets/05_Media_Center/PDFs/englisch/Industry_4.0_EN_web-g.pdf.

Tay, J. C. and Ho, N. B. (2008) ‘Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems’, *Computers and Industrial Engineering*, 54(3), pp. 453–473. doi: 10.1016/j.cie.2007.08.008.

Tsai, S. C. and Fu, S. Y. (2014) ‘Genetic-algorithm-based simulation optimization considering a single stochastic constraint’, *European Journal of Operational Research*. Elsevier B.V., 236(1), pp. 113–125. doi: 10.1016/j.ejor.2013.11.034.

Vieira, G. E., Herrmann, J. W. and Lin, E. (2003) ‘Rescheduling Manufacturing Systems : a Framework of Strategies , Policies , and Methods’, pp. 39–62.

Waschneck, B., Altenmüller, T., Bauernhansl, T. and Kyek, A. (2016) ‘Production Scheduling in Complex Job Shops from an Industrie 4 . 0 Perspective : A Review and Challenges in the Semiconductor Industry’.

Wee, D., Kelly, R., Cattel, J. and Breunig, M. (2015) ‘Industry 4.0 - how to navigate digitization of the manufacturing sector’, *McKinsey & Company*, pp. 1–62. doi: 10.1007/s13398-014-0173-7.2.

Wilbrecht, J. K. and Prescott, W. B. (1969) ‘The Influence of Setup Time on Job Shop Performance’, *Management Science*, 16(4), pp. B274–B280. doi: 10.1287/mnsc.16.4.B274.

Appendix A: Mean Performance of the methods applied to the benchmark instances for both static and dynamic approaches

Results						
Method	0	0,2	0,4	0,6	0,8	1
FCFC/STAT	1,208	1,226	1,339	1,488	1,622	1,703
SPT/STAT	1,220	1,266	1,380	1,491	1,705	1,860
LPT/STAT	1,328	1,373	1,504	1,666	1,785	1,967
LSD/STAT	1,191	1,222	1,355	1,503	1,616	1,799
MWF/STST	1,120	1,145	1,244	1,397	1,493	1,692
LTR1/STAT	1,145	1,169	1,285	1,419	1,576	1,722
MOR/STAT	1,168	1,194	1,311	1,454	1,562	1,717
FCFC/DYN	1,208	1,221	1,266	1,331	1,414	1,518
SPT/DYN	1,220	1,237	1,283	1,348	1,422	1,525
LPT/DYN	1,328	1,348	1,403	1,481	1,564	1,677
LSD/DYN	1,191	1,207	1,249	1,314	1,393	1,501
MWF/DYN	1,120	1,139	1,185	1,251	1,333	1,436
LTR1/DYN	1,145	1,168	1,222	1,297	1,384	1,496
MOR/DYN	1,168	1,179	1,223	1,285	1,366	1,473
Average Results						
STAT	1,197	1,228	1,345	1,488	1,623	1,780
DYN	1,197	1,214	1,262	1,330	1,411	1,518

Appendix B: Mean Performance of the methods applied to the real-world instances for both static and dynamic approaches

Results						
Method	0	0,2	0,4	0,6	0,8	1
FCFC/DYN	1,4441	1,4622	1,4591	1,4760	1,5020	1,5316
SPT/DYN	1,5011	1,5251	1,5400	1,5521	1,5853	1,6235
LPT/DYN	1,4960	1,5148	1,5065	1,5316	1,5414	1,5906
LSD/DYN	1,4307	1,4623	1,4434	1,4714	1,4882	1,5119
MWF/DYN	1,4428	1,4504	1,4489	1,4649	1,5007	1,5144
LTR/DYN	1,5025	1,5088	1,5119	1,5157	1,5521	1,5682
MOR/DYN	1,4402	1,4376	1,4336	1,4403	1,4607	1,4932
SSPT/DYN	1,4441	1,3727	1,3783	1,4001	1,4087	1,4441
MMS/DYN	1,0123	1,0308	1,0657	1,1018	1,1532	1,1480
SNSPT/DYN	1,1583	1,1676	1,2029	1,2119	1,2695	1,3058
GA/DYN	1,3898	1,4313	1,4225	1,3776	1,4698	1,5411
GH/DYN	1,0471	1,0608	1,1238	1,1063	1,1660	1,2488
CH/DYN	1,0455	1,0562	1,0799	1,1190	1,1453	1,2387
FCFC/STAT	1,4441	1,4427	1,4457	1,4648	1,4839	1,5626
SPT/STAT	1,5011	1,5103	1,5203	1,5590	1,6128	1,6099
LPT/STAT	1,4960	1,5319	1,5652	1,6061	1,6596	1,7247
LSD/STAT	1,4307	1,4450	1,4661	1,4922	1,5106	1,5699
MWF/STAT	1,4428	1,4459	1,4691	1,4801	1,5106	1,5322
LTR/STAT	1,5025	1,5293	1,5577	1,5995	1,6462	1,6649
MOR/STAT	1,4402	1,4466	1,4417	1,4415	1,4708	1,4971
SSPT/STAT	1,4441	1,4783	1,4569	1,5028	1,5345	1,5876
MMS/STAT	1,0123	1,0345	1,0837	1,1149	1,1489	1,1998
SNSPT/STAT	1,1583	1,2008	1,2106	1,2410	1,2984	1,3865
GA/STAT	1,3898	1,4764	1,4972	1,5265	1,5458	1,5936
GH/STAT	1,0455	1,0653	1,1134	1,1494	1,2012	1,2542
CH/STAT	1,0493	1,0666	1,1057	1,1327	1,2015	1,2955
Average Results						
DYN	1,3740	1,3909	1,3970	1,4055	1,4412	1,4862
STAT	1,3351	1,3595	1,3795	1,4085	1,4481	1,4983

Appendix C: Mean makespan and production capacity for the benchmark instances with setup level generated of 1

Results Makespan

Method	0	0,2	0,4	0,6	0,8	1
FCFC	1,591	1,598	1,617	1,646	1,669	1,728
SPT	1,620	1,687	1,704	1,736	1,763	1,815
LPT	1,698	1,711	1,732	1,764	1,797	1,854
LSD	1,609	1,615	1,634	1,657	1,687	1,748
MWF	1,493	1,503	1,526	1,555	1,588	1,641
LTR	1,524	1,536	1,560	1,598	1,639	1,701
MOR	1,557	1,555	1,567	1,590	1,618	1,666
SSPT	1,648	1,634	1,653	1,676	1,708	1,756
MMS	1,513	1,518	1,535	1,558	1,588	1,644
SNSPT	1,622	1,634	1,652	1,682	1,709	1,766
GA	1,550	1,594	1,620	1,648	1,680	1,742
CH	1,584	1,601	1,620	1,649	1,682	1,740
GH	1,590	1,605	1,621	1,652	1,683	1,742

Results Production Capacity

Method	0	0,2	0,4	0,6	0,8	1
FCFC	0,680	0,679	0,680	0,677	0,673	0,673
SPT	0,677	0,675	0,673	0,673	0,672	0,669
LPT	0,675	0,673	0,672	0,672	0,671	0,669
LSD	0,678	0,677	0,675	0,674	0,670	0,670
MWF	0,677	0,674	0,673	0,675	0,672	0,671
LTR	0,677	0,677	0,674	0,675	0,675	0,672
MOR	0,679	0,680	0,677	0,677	0,676	0,670
SSPT	0,675	0,675	0,675	0,674	0,671	0,667
MMS	0,681	0,680	0,679	0,676	0,676	0,673
SNSPT	0,678	0,675	0,675	0,674	0,671	0,670
GA	0,680	0,678	0,676	0,677	0,673	0,671
CH	0,679	0,679	0,679	0,677	0,676	0,674
GH	0,681	0,678	0,678	0,675	0,675	0,673

Appendix D: Mean makespan and production capacity for the benchmark instances with setup level generated of 0,75

Results Makespan

Method	0	0,2	0,4	0,6	0,8	1
FCFC	1,556	1,565	1,590	1,621	1,658	1,722
SPT	1,598	1,603	1,627	1,662	1,704	1,772
LPT	1,659	1,679	1,704	1,748	1,790	1,863
LSD	1,568	1,577	1,599	1,630	1,668	1,737
MWF	1,450	1,466	1,492	1,529	1,569	1,634
LTR	1,482	1,501	1,530	1,576	1,625	1,694
MOR	1,508	1,518	1,537	1,565	1,604	1,663
SSPT	1,588	1,596	1,617	1,650	1,687	1,750
MMS	1,474	1,486	1,507	1,538	1,577	1,642
SNSPT	1,594	1,599	1,619	1,656	1,691	1,764
GA	1,503	1,518	1,536	1,584	1,617	1,677
CH	1,568	1,571	1,591	1,630	1,672	1,742
GH	1,567	1,575	1,595	1,628	1,674	1,744

Results Production Capacity

Method	0	0,2	0,4	0,6	0,8	1
FCFC	0,705	0,706	0,704	0,703	0,697	0,697
SPT	0,706	0,704	0,703	0,702	0,698	0,696
LPT	0,700	0,699	0,699	0,695	0,691	0,692
LSD	0,709	0,704	0,703	0,701	0,697	0,697
MWF	0,700	0,700	0,701	0,699	0,699	0,697
LTR	0,703	0,702	0,702	0,699	0,697	0,696
MOR	0,706	0,707	0,703	0,702	0,699	0,696
SSPT	0,704	0,703	0,703	0,702	0,701	0,699
MMS	0,706	0,706	0,707	0,704	0,703	0,696
SNSPT	0,706	0,704	0,705	0,703	0,701	0,700
GA	0,705	0,706	0,704	0,703	0,698	0,697
CH	0,707	0,707	0,706	0,702	0,700	0,697
GH	0,709	0,706	0,706	0,703	0,699	0,699

Appendix E: Mean makespan and production capacity for the benchmark instances with setup level generated of 0,5

Results Makespan

Method	0	0,2	0,4	0,6	0,8	1
FCFC	1,512	1,530	1,554	1,596	1,645	1,722
SPT	1,547	1,559	1,589	1,632	1,681	1,767
LPT	1,642	1,651	1,682	1,734	1,793	1,871
LSD	1,526	1,532	1,553	1,598	1,647	1,732
MWF	1,413	1,428	1,461	1,505	1,558	1,632
LTR	1,436	1,461	1,501	1,555	1,611	1,694
MOR	1,465	1,478	1,502	1,542	1,588	1,664
SSPT	1,540	1,552	1,577	1,618	1,667	1,745
MMS	1,441	1,446	1,476	1,515	1,570	1,649
SNSPT	1,551	1,553	1,584	1,620	1,678	1,762
GA	1,499	1,543	1,568	1,618	1,667	1,752
CH	1,550	1,545	1,572	1,623	1,674	1,760
GH	1,537	1,552	1,578	1,623	1,680	1,758

Results Production Capacity

Method	0	0,2	0,4	0,6	0,8	1
FCFC	0,7433	0,7412	0,7374	0,7346	0,7301	0,7251
SPT	0,7398	0,7433	0,7408	0,7391	0,7376	0,7336
LPT	0,7334	0,7340	0,7269	0,7252	0,7231	0,7196
LSD	0,7457	0,7426	0,7410	0,7381	0,7349	0,7329
MWF	0,7371	0,7358	0,7370	0,7329	0,7311	0,7262
LTR	0,7366	0,7380	0,7359	0,7330	0,7278	0,7237
MOR	0,7420	0,7412	0,7395	0,7357	0,7329	0,7280
SSPT	0,7391	0,7415	0,7419	0,7389	0,7384	0,7329
MMS	0,7410	0,7412	0,7397	0,7371	0,7339	0,7296
SNSPT	0,7405	0,7429	0,7408	0,7403	0,7370	0,7330
GA	0,7420	0,7436	0,7390	0,7372	0,7331	0,7320
CH	0,7452	0,7411	0,7395	0,7334	0,7323	0,7267
GH	0,7475	0,7446	0,7403	0,7369	0,7320	0,7272

Appendix F: Mean makespan and production capacity for the benchmark instances with setup level generated of 0,25

Results Makespan

Method	0	0,2	0,4	0,6	0,8	1
FCFC	1,469	1,476	1,509	1,565	1,636	1,734
SPT	1,485	1,499	1,535	1,592	1,662	1,759
LPT	1,599	1,607	1,651	1,712	1,789	1,894
LSD	1,451	1,467	1,505	1,561	1,626	1,730
MWF	1,358	1,373	1,417	1,472	1,546	1,636
LTR	1,389	1,407	1,455	1,523	1,600	1,701
MOR	1,419	1,427	1,459	1,513	1,579	1,676
SSPT	1,484	1,493	1,524	1,579	1,642	1,743
MMS	1,395	1,403	1,439	1,494	1,562	1,661
SNSPT	1,486	1,503	1,533	1,587	1,654	1,754
GA	1,451	1,483	1,522	1,584	1,658	1,766
CH	1,493	1,510	1,544	1,603	1,669	1,777
GH	1,499	1,510	1,544	1,603	1,669	1,777

Results Production Capacity

Method	0	0,2	0,4	0,6	0,8	1
FCFC	0,680	0,679	0,680	0,677	0,673	0,673
SPT	0,677	0,675	0,673	0,673	0,672	0,669
LPT	0,675	0,673	0,672	0,672	0,671	0,669
LSD	0,678	0,677	0,675	0,674	0,670	0,670
MWF	0,677	0,674	0,673	0,675	0,672	0,671
LTR	0,677	0,677	0,674	0,675	0,675	0,672
MOR	0,679	0,680	0,677	0,677	0,676	0,670
SSPT	0,675	0,675	0,675	0,674	0,671	0,667
MMS	0,681	0,680	0,679	0,676	0,676	0,673
SNSPT	0,678	0,675	0,675	0,674	0,671	0,670
GA	0,680	0,678	0,676	0,677	0,673	0,671
CH	0,679	0,679	0,679	0,677	0,676	0,674
GH	0,681	0,678	0,678	0,675	0,675	0,673

Appendix G: Mean makespan performance of the method evolved using genetic programming

	cv					
Set of instances	0	0,2	0,4	0,6	0,8	1
Real-world	1,183	1,193	1,184	1,188	1,184	1,192
SL = 1	1,450	1,451	1,470	1,494	1,533	1,578
SL = 0,25	1,363	1,374	1,410	1,466	1,537	1,635