FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Industry 4.0 - Shop-Floor Negotiation

Eduardo Miguel Bastos Leite

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ana Paula Rocha

July 3, 2019

Industry 4.0 - Shop-Floor Negotiation

Eduardo Miguel Bastos Leite

Mestrado Integrado em Engenharia Informática e Computação

July 3, 2019

Abstract

Today's manufacturing is facing unprecedented challenges to meet the ever-growing customer expectations for high-quality, fast-delivered, customizable products. But with the emergence of the Industry 4.0 paradigm, industries now have a powerful ally to face these modern challenges. In this dissertation, we explore and discuss the current state of the main concepts that comprise the Industry 4.0 universe. Principally, how they can be used to solve a modern version of the job-shop scheduling problem, that has been stemming the curiosity of academics since the twentieth century. With this, we aim to improve factory efficiency by promoting better use of factory resources, through the advancements on the scheduling strategies while bridging the gap between academics and industry. In this dissertation, we explore actual and historical strategies researchers used throughout the years to solve this problem, complemented with our views on what should be a modern interpretation of the problem. This dissertation also puts forward an agent-based solution that allows partial or total rescheduling of job shop orders using dispatch rules and a genetic algorithm. Towards the end, we present and discuss the results obtained, also stating some of the future work that we believe would complement our project.

Resumo

Hoje em dia a indústria está a enfrentar desafios sem precedentes para atender às crescentes expectativas dos clientes por produtos personalizáveis de alta qualidade, entregues rapidamente. Mas com o surgir do paradigma da Indústria 4.0, as indústrias agora têm um poderoso aliado para enfrentar esses desafios modernos. Nesta dissertação, exploramos e discutimos o estado atual dos principais conceitos que compõem o universo da Indústria 4.0. Principalmente, como os podemos utilizar para resolver uma versão moderna do problema de escalonamento de ordens de fabrico, que tem vindo a despertar a curiosidade dos académicos desde o século XX. Com isso, pretendemos melhorar a eficiência das fábricas, promovendo uma melhor utilização dos seus recursos, através dos avanços nas estratégias de escalonamento, simultaneamente fazendo a ponte entre os académicos e a indústria. Nesta dissertação, exploramos estratégias actuais e históricas utilizadas pelos investigadores ao longo dos anos para resolver este problema, complementadas com nossas visões sobre o que deveria ser uma interpretação moderna do problema. Esta dissertação também apresenta uma solução baseada em agentes que permite o reescalonamento parcial ou total de pedidos de ordens de fabrico utilizando regras de despacho e um algoritmo genético. No final, apresentamos e discutimos os resultados obtidos, indicando também algum do trabalho futuro que acreditamos que complementaria o nosso projecto.

Acknowledgements

ıI want to thank my supervisor Ana Paula Rocha, for the knowledge she shared with me, and for all the dedication in answering my questions and help me improve this dissertation.

I would also like to thank my supervisors at Critical Manufacturing, Pedro Pereira, and Samuel Rodrigues, for patiently guiding me through all the concepts I had to learn.

To Jose Pedro Silva, Luís Pinho, and Micael Queiroz, three coworkers that I admire, for helping me remove every obstacle in this dissertation.

To Heldér Ferreira, for having infinite patience to help me build my queries.

To my friends and family, for their constant support.

And last but not least to my girlfriend Andreia Rodrigues, for all the dedication in helping me complete this dissertation.

Eduardo Leite

"If you fail to plan, you are planning to fail"

Benjamin Franklin

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

ACC	Agent Communication Channel
ACO	Ant Colony Optimization
AI	Artificial Intelligence
AID	Agent Identifiers
AMS	Agent Management System
B2B	Business-to-Business
BnB	Branch and Bound
BS	Beam Search
CC	Cloud Computing
CM	Cloud-Manufacturing
CPPS	Cyber Physical Processing System
CPS	Cyber Physical System
DF	Directory Facilitator
DPC	Delay Precedence Constraints
EA	Evolutionary Algorithms
EC	Edge Computing
EDD	Earliest Due Date
FIPA	Foundation for Intelligent Physical Agents
FJSP	Flexible Job-Shop Scheduling Problem
FSP	Filtered Beam Search
FSSP	Flow-Shop Scheduling Problem
GA	Genetic Algorithms
HTTP	HyperText Transfer Protocol
IIoT	Industrial Internet of Things
IoT	Internet of Things
ISIS	Intelligent Scheduling and Information System
JADE	Java Agent Development Framework
JSON	JavaScript Object Notation
JSP	Job-Shop Problem
JSSP	Job-Shop Scheduling Problem
MES	Manufacturing Execution System
NN	Neural Networks
OPIS	Opportunistic Intelligent Scheduler
PFJSP	Permutation Flow Shop Scheduling Problem
SA	Simulated Annealing
TS	Tabu Search

Chapter 1

Introduction

1.1 Context/Background

The ever-growing customer expectations about product delivery quality are increasing the need for companies to modernize themselves in order to stay ahead in the highly competitive environment of industrial production. The search for very specific and customized products, with unique characteristics, is an idea that challenges the traditional mass production concepts. To overcome this challenge, it is necessary for each order with its unique specifications and requirements to be represented in the shop-floor, negotiating with the manufacturing execution system so that it can plan resource allocation in the best possible way [GM16, WYT15, ROL18].

The act of allocating finite resources to activities in a set period of time is called scheduling and it has a direct impact on the factory's ability to deliver products with a limited amount of time and money. Because of the innate dynamic and distributed nature of shop-floor production, traditional centralized scheduling solutions have some difficulty coping with unpredictable real-time events such as machine failures, requirements change, and new orders. This happens because a centralized approach can only solve the problem as a whole, not being able to reschedule portions of the problem [WYT15].

With the new Industry 4.0 concepts, products are getting smarter, and the factories ever more digital. Instead of using a centralized methodology, we propose the use of distributed artificial intelligent agents that can efficiently solve this issue, by having them advocate for their customers, negotiating their processes, required equipment and quality tolerances, basically their journey throughout the shop-floor.

In order for this to happen, it is necessary to have well-defined protocols for agents to negotiate between them and with the manufacturing execution system. Nowadays, there is a big variety of protocols being used across multiple subjects, some of which are able to provide the needed decentralized approach to the scheduling problem enabling the possibility to better react to real-time events and to provide partial solutions [WcXx09].

1.2 Aim and Goals

The aim of this dissertation is to propose a decentralized, multi-agent based, service-oriented solution to the scheduling problem, achieving a better response time to customer orders and requirements by promoting better use of resources in the shop-floor, while providing a better tolerance to failure and unexpected events and the ability to achieve real-time solutions. Another goal is to decrease the scheduling algorithm execution time, this is possible due to the state space being smaller if the problem is broken down into multiple smaller problems solved independently by different solvers (agents).

By doing so we expect to provide a solution to this problem that can have a real impact on the industry and the economy, by having better resource management, factories will be able to provide more product customization delivering a unique experience to his customers while meeting budgets and deadlines. The lower production cost can also change the market by making products cheaper and give companies more margin to raise salaries and promote skillful employees [KGG18].

1.3 Document Structure

In Chapter 2 we talk about the context of the dissertation, namely Industry 4.0, schedule optimization and the JADE framework, proving the necessary definitions and exploring some of the existing literature about these topics, with a strong emphasis on the optimization of schedules through a decentralized, service based, multi-agent environment. Then in Chapter 3 we describe the problem being addressed, first the general case and then specifically the Critical Manufacturing case. In Chapter 4 we talk about the implementation of our solution to the problem, presenting an early architecture draft, and in Chapter 5 we present the results. Lastly in Chapter 6 we present the final remarks and discuss possible directions for future work. Appendix A includes a paper to be submitted to the International Conference on Advanced Information Systems 32nd Engineering (CAiSE'20),

Chapter 2

Literature Review

In this Chapter, we aim to provide the reader with the knowledge needed about the context of this dissertation, describing important concepts and exploring some of the existent work and techniques in relevant fields. We start in Chapter 2.1 by looking at the current state of the industry, defining and exploring some of the central technologies of the industry 4.0 paradigm. Then in Chapter 2.2, we address the Job-Shop Scheduling problem (JSSP), by looking at some of the most recent techniques used to solve it. While at the same time providing a historical overview of how researchers addressed this problem throughout the years. Finally, in Chapter 2.3, we briefly discuss the JADE framework that we used in this dissertation.

2.1 Industry 4.0

In this Section, we talk about industry 4.0, briefly explaining its origins and presenting the factors that motivated it as well as defining some of the main paradigms that are associated with it, namely Cyber-Physical Systems (CPS), Internet of Things (IoT), Cloud Computing (CC) and Artificial Intelligence (AI). In the end, we conclude with a note about Manufacturing Execution Systems (MES) and the impacts of the revolution.

2.1.1 Origins

The industrial sector has benefited from technological advancements ever since it first appeared. Nonetheless, it is essential to note that these improvements don't always happen gradually, sometimes great discoveries or ideas can create the conditions for a rapid positive change to the existing industrial paradigm in a short time, a phenom called industrial revolution. These revolutions cause profound social and economic changes, and so, it is of utmost importance that we understand and document them [LHW17, Sch17, PAG18].

Looking into the past, we can identify three moments in history where these revolutions happened (as illustrated in figure 2.1). The First Industrial Revolution began in England and quickly spread to the rest of the world. The main idea behind it was the use of water and steam to power

machines and factories. Then the Second Industrial Revolution was a combination of the discovery of electricity and the subsequent appearance of mass production factories capable of separating their work into specific areas. Lastly, the Third Industrial Revolution happened with the advancements of electronics and information systems that greatly enhanced the ability of factories to automate their industrial processes [LHW17, Sch17, LM18].

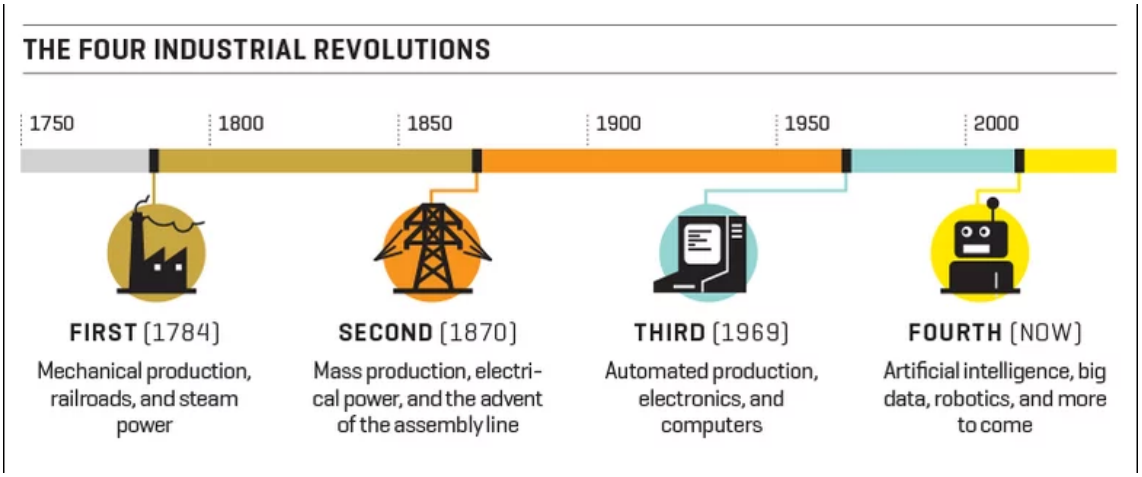

Figure 2.1: The four industrial revolutions. (Source: World Economic Forum)

2.1.2 Definition

Today our industries are facing high pressure to adapt to the principles of a new revolution. Motivated not only by the ever-growing expectations of customers for highly customizable, cheap, fast delivered and high-quality products but also by the need to improve to stay ahead in the strongly competitive environment of the industrial sector, the industry is looking to modernize itself, striving for a more efficient and automated future [LHW17, Sch17, KGG18].

The name commonly given to this fourth revolution is Industry 4.0. The term first appeared in 2011 at the Hanover Fair, proposed by the German government and it is currently a buzzword in several fields of study from engineering to economics [LM18, LHW17, PAG18, DH14].

It is common to identify the Internet of Things and Cyber-Physical Systems paradigms as pillars of Industry 4.0, which unlike previous revolutions centers itself in the digital world and its connection to the physical world instead of happening almost exclusively in the physical world [KGG18, LHW17, Sch17, LM18, BB19, LOCS19, SKK19]. Other concepts that usually appear intertwined with Industry 4.0 are Cloud Computing [LHW17, KGG18, LM18, PAG18, BB19, CBCJO19, LOCS19, SKK19] and Artificial Intelligence [PAG18, Sch17, LHW17]. We will explore each of these concepts in its respective Subsection.

2.1.3 Cyber-Physical Systems

One of the fundamental aspects of Industry 4.0 is the digitalization of physical systems and processes. And because this is the goal of Cyber-Physical Systems, the connection between the two

becomes self-evident. Some authors even go as far as saying the two can be considered synonyms. While others feel the need to draw a clear distinction between the two, claiming that CPSs are only part of the Industry 4.0 concept [LHW17, DH14, MZ16].

Following the popularization and advancement of Embedded Systems, Cyber-Physical Systems are now further enhancing the relationship between the digital and the physical. In a Cyber-Physical System, a collection of computational objects work together to offer a digital view of a set of real-world entities and their interactions (as illustrated in figure 2.2). This digital representation maintains consistency with the physical world through the rapid and constant exchange of information, recalculating its output every time the input changes. It is important to note that Cyber-Physical Systems are more about the interactions between the digital and physical world than any of them particularly. What makes these systems particularly useful is the fact that they provide the user with real-time monitoring and control of a system, or in the case of Industry 4.0, of a factory. A CPS that is specific to industrial use is called a Cyber-Physical Production System (CPPS) [MKB+16, PZL12].

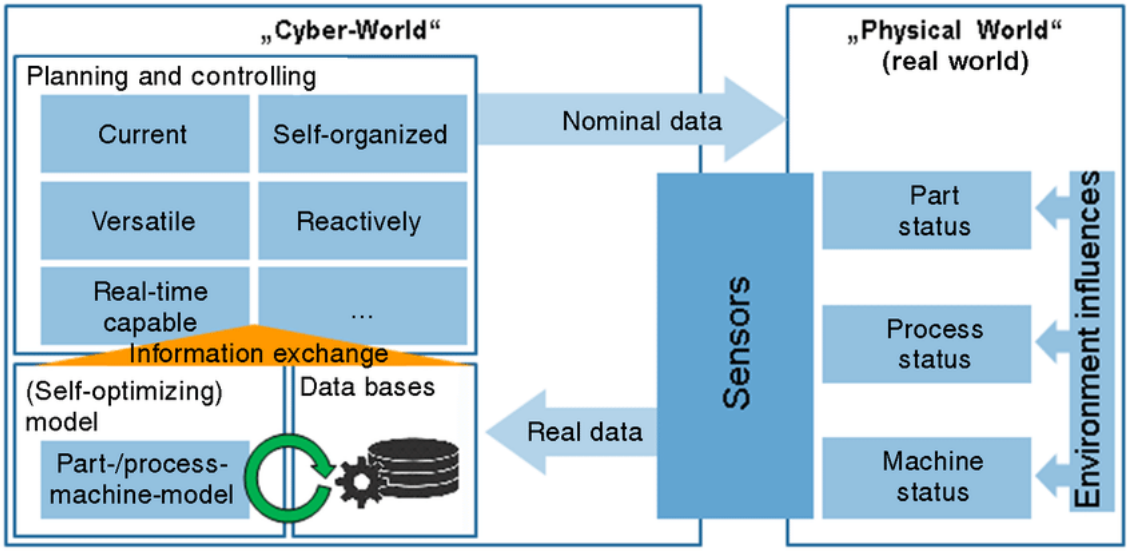

Figure 2.2: Cyber Physical System. (Source: [IBH+16])

As of today, CPSs continue to be a hot field of study because of its many industrial and non-industrial applications. In critical situations, the enhanced rate at which we can access information can make a huge difference. For instance, in the healthcare and autonomous driving fields, it can be the difference between an accident happening or not. In the context of an industry, real-time information can be used for condition-based maintenance or shop-floor scheduling (this paper from 2018 [MV18] is an excellent example that tries to use CPS in combination with cloud computing to achieve both) greatly improving existing systems that are subject to delays [PZL12].

In the Industry 4.0 field, the concepts of CPS and Internet of Things (IoT) are often linked. The IoT concept is discussed in the following Section.

2.1.4 Internet of Things

As previously mentioned, the Internet of Things paradigm and its industrial branch Industrial Internet of Things (IIoT) (illustrated in figure 2.3) are vital parts of the Industry 4.0 concept. Its main goal is the collection and communication of real-time information between physical objects and the digital world. What makes this possible is the existence of multiple devices, fixed or mobile, connected via the internet. In a smart-factory, shop-floor devices (particularly machines) are all connected using the internet. IoT devices collect data through sensors and then use the available connectivity to transfer it to the network. This data usually refers to a machine's internal state, from internal temperature and rotation speed to power consumption and vibration. Even though CPS and IoT are related, they meet different needs. While CPSs are more concerned about the interactions between the physical and digital world, and its representation (as illustrated in figure 2.4). IoT is mainly concerned about data collection and availability. Similar to CPS, IoT is a popular field of study. Researchers are still discovering new applications for IoT, with a big focus on the uses of IoT for maintenance [CAG+18, KYK18].

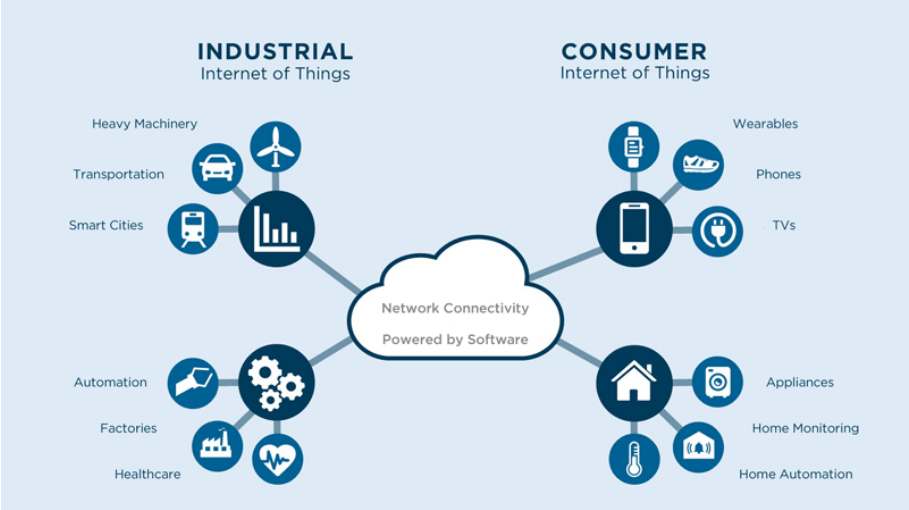

Figure 2.3: IoT and IIoT. (Source: [mde16])

2.1.5 Cloud Computing

Another important technology that is often associated with Industry 4.0 is Cloud Computing. With the implementation of the CPS and IoT paradigms, the amount of information that is required to be stored and managed drastically increases. And so, it is necessary to find a way to save this newly acquired data, in a centralized way, promoting data availability and consistency. By using Cloud Computing, it is possible to meet these goals while keeping storage costs to a minimum (these and other benefits are illustrated in figure 2.5).

CC solutions usually offer pay-as-you-go plans, with easy access to scaling control and across multiple available platforms. This way, a company only pays for what it needs. Furthermore, CC solutions can enable robust complex and distributed architectures, as it is the case of Business-to-Business (B2B) architectures that provide negotiation and cooperation ground for manufacturers

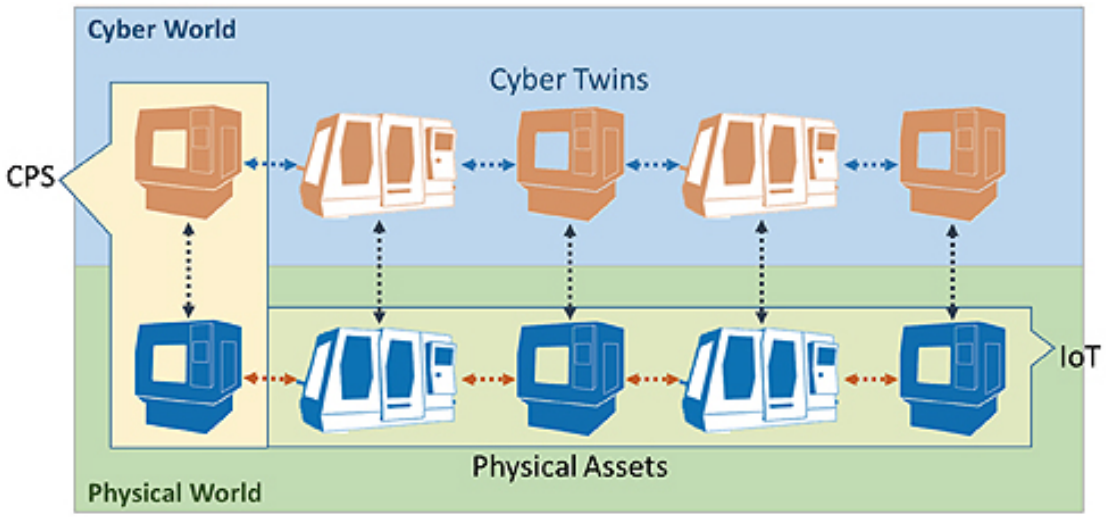

Figure 2.4: Cyber Physical Systems and Internet of Things. (Source: [Wor15])

and retailers. Moving complex interactions with the customer to the web, increases those processes efficiency [MV18, Xu12, SPKG18].

One of the branches of Cloud Computing that can also benefit the industry is Cloud Manufacturing (CM). By having the data centralized in the cloud, it is possible to coordinate extensive processes that involve distributed facilities. It is important to note that the main concern regarding Cloud Computing and Manufacturing is security. Ensuring data confidentiality and security is a crucial aspect that will impact how industry stakeholders will look at these technologies [Xu12].

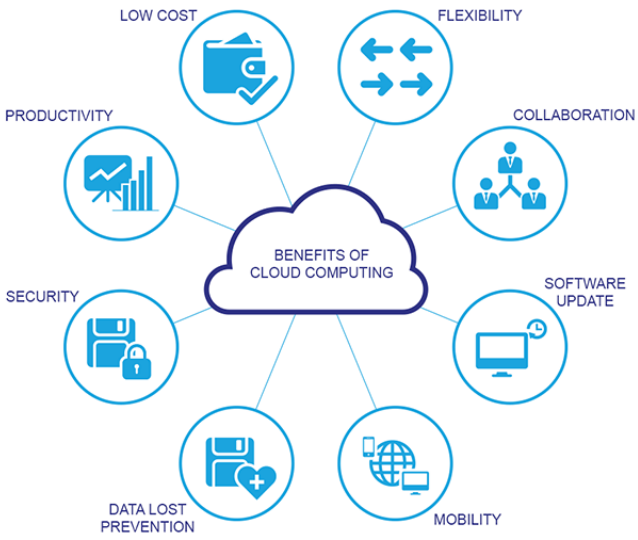

Figure 2.5: Benefits of Cloud Computing. (Source: [vis])

More recently, an extension of Cloud Computing called Edge Computing (EC) is gaining the attention of the Industry. This technology aims to solve existing limitations by moving the computational efforts to the "edge" of the system. In practice, this means that some of the work per-

formed by the centralized cloud system now gets done by local devices. By doing so, it is possible to mitigate some security concerns while diminishing bandwidth costs and considerably reducing latency. Because several industrial processes require real-time responses, edge computing can serve as a significant enhancement to the existing technological stack [SCAC+19, YLH+18].

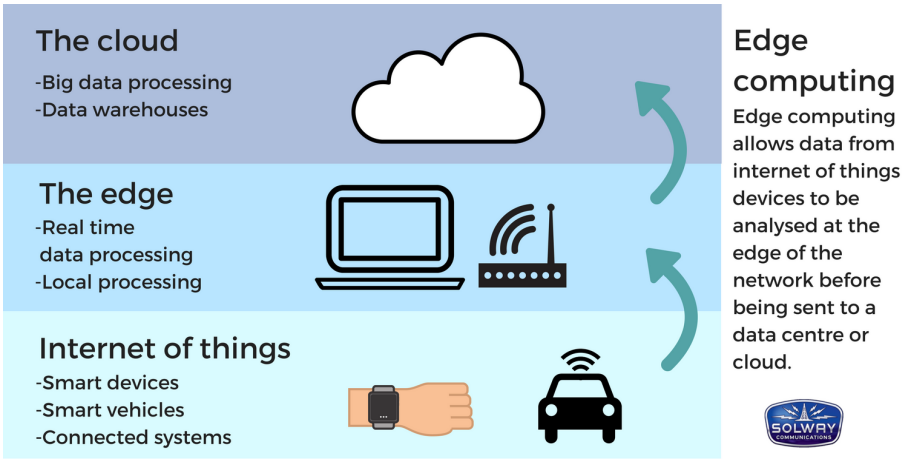

Figure 2.6: Edge Computing. (Source: [sol])

2.1.6 Artificial Intelligence

The last industry 4.0 technology that we will talk about is Artificial Intelligence. It is important to note that although every enunciated technology can impact scheduling, AI is the one that is more directly related to the problem we will address in Chapter 3.

Artificial Intelligence is the science that studies techniques and algorithms that can give machines, particularly computers, human-like behaviour and intelligence. This intelligence is defined as the capability of solve a problem, make a decision, or even learning with some or total autonomy. Being able to introduce intelligence in non-humans, AI plays an essential role in any industrial advancement and as a consequence in Industry 4.0. Because it's definition is quite broad, many technologies fall in the boundary of AI (as illustrated in figure 2.7). And due to their clear benefit to the industry, research in these fields is of utmost importance.

AI can be used in many fields. AI in image and speech recognition can enhance our ability to provide inputs to the digital world. In the future, it is possible that computers can use AI to understand our human communication, greatly heightening the quality of our interactions with cyber systems. Machine learning can also make it possible for machines to perform complex tasks with a high degree of autonomy, thus increasing work efficiency, and in some cases reducing the danger to human employees. Due to its ability to process vast amounts of information in a small amount of time, AI can also help humans make smart decisions. This improved decision making can reduce manufacturing costs while promoting efficiency and allowing for a more customized experience of the customers. One of the fields in which AI can help make decisions is the scheduling of customer orders, which is the major theme of this dissertation [Lu19, QMQ+19, PME19].

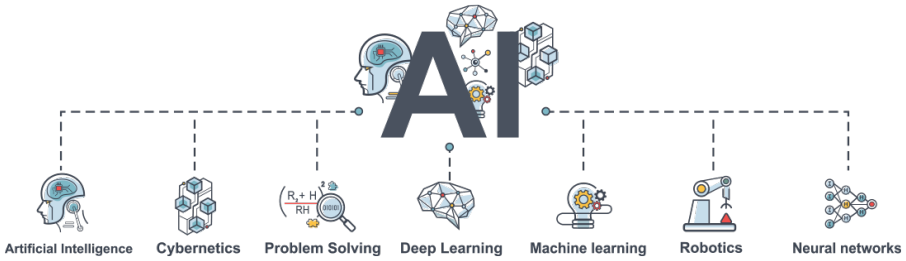

Figure 2.7: AI Technologies. (Source: [Kin19])

2.1.7 Manufacturing Execution System

In this Subsection, we will briefly define what is a Manufacturing Execution System (MES), and discuss its relationship with the Industry 4.0. Later, in Chapter 4 (Implementation), we will describe how we have used data from an MES to build our solution.

Critical Manufacturing describes an MES as "...an information system that drives the execution of manufacturing operations" [Man]. Through the use of an MES, it is possible to store all the relevant information for monitoring and reporting. This information can then help both decision makers, via decision support systems, and shop-floor workers, by presenting the necessary information for their role. Figure 2.8 illustrates some of the short and long term benefits provided by the MES. Critical Manufacturing further describes an MES by clarifying the difference between an MES and a traditional ERP. According to them, an MES is "... the ideal choice for a complex production process with multiple variations and a massive number of transactions". Whereas an ERP "... is generally designed to support a homogeneous process with business operating information" (Illustrated in figure 2.9) [Man].

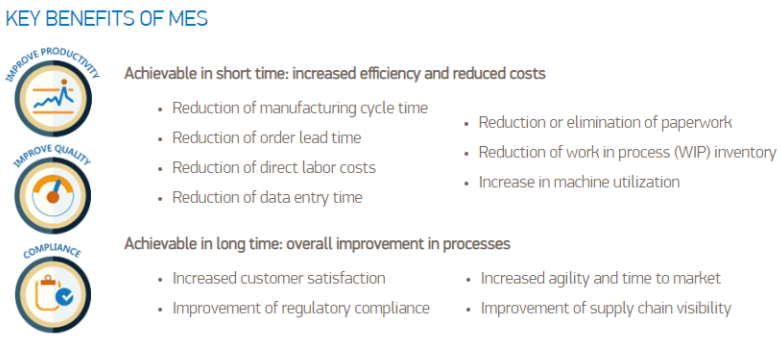

Figure 2.8: Key benefits of MES. (Source: [Man])

2.1.8 Social Impacts

Like any industrial revolution, industry 4.0 has big social impacts, not only for manufacturers but also to the consumer. With the increased automation level, enhanced monitoring and better planning, it is possible to reduce the manufacturing times and costs, allowing better margins for the suppliers and cheaper products for the buyers. Another important benefit is the increased

Figure 2.9: MES vs ERP. (Source: [Man])

customization of the products, by having a better organization and networking between production units, it is possible to provide customers with a more unique experience, an idea that was traditionally conflicted with mass production [KGG18, LHW17].

2.2 Shop-Floor Optimization

In this Section, we talk specifically about shop-floor optimization. Starting with a small introduction to shop-floor and optimization under Industry 4.0, and then moving to the Job Shop Scheduling (JSP) problem that this dissertation aims to tackle. We will look at how the current literature defines this problem and what variants of the simple problem exist. We will then explore existing techniques to solve different versions of the problem, grouped by their strategy, always accompanied by a description of its advantages, disadvantages, and limitations. Our goal is that by the end of this Section, the reader can clearly understand the problem at hands as well as the main strategies that can be used to solve it. Providing a solid base of knowledge that will help comprehend our interpretation and solution for the problem.

2.2.1 Definition

The term Job-Shop refers to the areas in a manufacturing facility where the work occurs. It's where factory workers, generally with the aid of machines and other tools, complete jobs that move materials along their production cycle. This process is what adds value to the factory input, and it is a core element of any manufacturing facility. However, because there is a finite number of human and mechanical resources, it is of great importance that the distribution of jobs among these resources is as efficient as possible. Because of the growing product customization requirements and due to the massive amount of information made available by the Industry 4.0 technologies, this task is now more complex than ever [WYT15, KGG18].

To this act of allocating finite resources to a set of tasks, we call scheduling. If we are referring to jobs in a shop-floor, we call it Job-Shop Scheduling (JSP). Modern smart-factories that operate

under the Industry 4.0 principles must be able to leverage the available real-time information to make informed decisions on how to schedule or reschedule factory jobs. Since each factory is different and has different goals and requirements, the Job-Shop Problem has many different formulations and solving techniques. In the next Subsections, we explore these techniques by looking at some of the existing work in this area. Later in Chapters 3 and 4, we will explain our formulation of the problem and corresponding solution [WYT15, ZDZ+19].

2.2.2 Exact Methods

The first proposed solutions for the job-shop scheduling problem aim to provide globally optimal solutions through the use of deterministic methods. Although these methods can provide the best solutions, except for specific versions of the problem, these solutions don't scale very well rapidly becoming unfit for a real-world scenario. But as Industry 4.0 makes industrial systems more distributed, it is possible that these methods can be used once again to solve local sub-problems or the whole JSP. For this to happen, it is necessary to understand how we can adapt these techniques to a modern day scenario [ZDZ+19, SM98].

2.2.2.1 Efficient Rule

The first efficient rule method was proposed by Johnson in 1954 [Joh54]. In his paper, Johnson introduces a set of rules that allowed for an optimal allocation of jobs under certain conditions (illustrated in table 2.1). These rules can achieve the optimal by minimization of the makespan, requiring only input of processing times for each job in each machine (illustrated in table 2.2). We describe this process in algorithm 1. Although being able to get optimal solutions in a reasonable time, because of the various preconditions it requires, the Johnson rules are unfit for nearly any real-life scenario. Nonetheless these rules are identified as one of the main driving factors of research in this domains, being the first paper to identify makespan minimization as a goal [ZDZ+19, SM98, Joh54].

Johnson Rule Requirements
Non variable processing time
All jobs need to be processed in the same order (first machine A then machine B)
There is no priority hierarchy between jobs

Table 2.1: Johnson Rule Requirements

Job	**Machine A**	**Machine B**
Job 1	5	3
Job 2	1	5
Job 3	2	3

Table 2.2: Johnson Rule Input Example

After Johnson published his work, a series of other researches put forward their own efficient rules [Ake56, Jac56, HA82]. Further research focused on proving that for instances of the

Algorithm 1: Johnson Rules Algorithm

```
Result: Optimally Ordered Jobs
MachineCount = 2;
JobCount = input.JobCount;
ProcessingTimes[JobCount][MachineCount] = input.ProcessingTimes;
ProcessingOrder[JobCount] = new Array[];
while ProcessingTimes.HasElements() do
    Job job = ProcessingTimes.GetLowestValue();
    if job.isMachineA() then
        ProcessingOrder.InsertAtStart(job);
    else
        ProcessingOrder.InsertAtEnd(job);
    end
    ProcessingTimes.DeleteLine(job);
end
```

JSP where machines and operations per job are more than two, no efficient rule method can be found because for these problems the complexity becomes NP-Hard [Bur84, GS78, GLLK79, KLHGRKB77]. For this reason, these rules cannot be used to solve most JSP problems. And so, the next exact methods we will look at, focus on eliminating solutions through constraints to be able to solve larger dimensions of the problem [ZDZ+19, SM98].

2.2.2.2 Mathematical Programming

Shortly after Johnson proposed his efficient rule method, in 1959 Wagner [MW59] discovered that is was possible to achieve optimal solutions with mathematical programming. Although being optimal, similarly to efficient rules, these algorithms don't scale well. Depending on the formulation, for large enough problems the solutions were of poor quality or the time necessary to run the algorithm was infeasible. In 1960 Manne [Man60] improved this solution by combining integer and linear programming with the linear programming for the constraints and the integer programming for decision variables. Despite being an improvement, this solution shared the same limitations as the original [ZDZ+19].

2.2.2.3 Branch and Bound

The number of different combinations in a classical job-shop scheduling problem with m machines and n jobs can be calculated using the expression $(n!)^m$. For this reason (as illustrated in table 2.3) the number of candidate solutions is rather large, even on small scenarios. And so, researches have started using branch and bound (BnB) methods to try to reduce this space, by discarding solutions that were known to not be optimal without the need to test them [ZDZ+19].

In 1965 G.H. Brooks and C.R. White [BW65], Z. A. Lomnicki [Lom65] and Edward Ignall and Linus Schrage [IS65], followed by G. B. McMahon and P. G. Burton in 1967 [MB67] and Subhash C. Sarin, Seokyoo Ahn and Albert B. Bishop in 1988 [SAB88], used branch and bound methods to optimize flow times in 2 and 3 machines. And then in 1994 Peter Brucker, Bernd Jurisch, and Bernd Sievers [BJS94] tested, BnB for the total completion time of a 10x10 problem.

Number of Machines	Number of Jobs	Solutions
1	1	1
1	3	6
1	5	120
2	2	4
2	3	36
2	5	14400
3	5	1728000

Table 2.3: JSP nm dimensionality examples

Other researchers used Branch and Bound methods to try to address the problem of meeting the jobs due date. First, in 1975 Graham McMahon and Michael Florian [MF75] proposed a solution to minimize the maximum lateness and then in 1985 Chris N. Potts and Luk Van Wassenhove [PVW85] for the single machine total weighted tardiness.

Despite being able to find globally optimal solutions in a lower amount of time than other exact methods, branch and bound techniques still weren't able to cope with the dimension of most real-life problems. In the next Subsections, we will look at some of the ways researches sacrificed the optimal solution in favor of strategies that were able to find good enough results in a more acceptable time.

2.2.3 Constructive Methods

In this Subsection, we will talk about construction methods. Similarly to exact methods, they rely on a set of rules to allocate jobs to machines. But as we will see, the main difference in construction methods is that their less strict rules allow them to create solutions faster by relaxing their requirements. Although this relaxation makes them unlikely to find optimal solutions, the speed at which they can provide a satisfying solution makes them interesting for Industry 4.0. It is important to note that one of the main scheduling features that can benefit from these methods is rescheduling. When a real-time situation that made the original scheduling unfeasible occurs, a quick rescheduling using constructive methods can avoid long periods with no schedule [ZDZ+19].

2.2.3.1 Dispatch Rules

As mentioned before, in every job shop scheduling problem, we have a set of jobs that have to be completed. Dispatch rules methods, aim to assign a priority to each of these jobs to then create a schedule around it (as described in 2). There are numerous examples of rules that can be used to distinguish jobs. Some of the most common are the job delivery date, the shortest processing time and the least total work remaining. It is worth noting that there isn't a particular set of rules that is superior for every case. And so, it is necessary to analyze each specific problem in its context to get the best results for the desired goals [RH99, ZABA15, PRTS16].

Dispatch rules research is still active and there are several papers that have been inspired by it. Some industry specific papers are the ones written by Yi-Feng Hung and Ing-Ren Chen in

Algorithm 2: Dispatch Rules Algorithm

```
Result: Dispatch Rules Allocation
Machines = input.Machines;
UnorderedJobs = input.UnorderedJobs;
OrderedJobs = new Array[] ;
while UnorderedJobs.HasElements() do
    Job job = UnorderedJobs.SelectHighestPriority(criteria); OrderedJobs.push(job);
end
while OrderedJobs.HasElements() do
    Job job = OrderedJobs.first(); Machine machine = Machines.GetAvailableEarliest();
      machine.queue.push(job); OrderedJob.delete(job);
end
```

1998 [HC98], and by O. Rose in 2001 [Ros01], who used dispatch rules to optimize flowtime in semiconductor manufacturing. Some of the most recent work uses multiple optimization criteria, usually accompanied by artificial intelligence learning techniques to tune the weight of each rule. Some example are the work of Helga Ingimundardottir and Thomas Philip Runarsson in 2011 [IR11] who combined supervised learning with dispatch rules. The work of M. H. Zahmani, B. Atmani, A. Bekrar and N. Aissani in 2015 [ZABA15] which combined multiple dispatch rules with Data Mining to achieve scheduling goal like reduce tardiness and flowtime. The work of Midhun Paul, R. Sridharan and T. Radha Ramanan in 2016 [PRTS16] that compared several dispatch rules in a multi-objective testing environment. And more recently the work of Rajan and Vineet Kumar in 2019 [RK19] that studies the application of different rules to different objectives and discusses the issue of scheduling under the presence of uncertain factors.

2.2.3.2 Insert Algorithm

Although most of constructive methods use Dispatch Rules, some authors found that insert algorithms can provide a better alternative [WW95, STW99]. In this approach instead of ordering by priority and then processed in that order, jobs are inserted into an empty or existing schedule, in the place that best fits the scheduling goals. In 1995, Frank Werner and Andreas Winkler [WW95] combined an insert algorithm with the beam search local algorithm to first construct an initial scheduling and then continuously improving it by re-inserting existing jobs, generating potential neighbours for the beam search. Then in 1999, Yuri Sotskov, Thomas Tautenhahn and Frank Werner [STW99] used an insertion algorithm to solve the problem of batch production, in which several jobs can be performed at the same time in a batch, with the restriction that the machine usually has to have a certain number of jobs to start processing the batch. Later in 2014 Yahong Zheng, Lian Lian and Khaled Mesghouni [LM14] published a paper compared various algorithms, including insert algorithms to factor the scheduling of maintenance operations. And more recently in 2019, A. Bekkar, G. Belalem and B. Beldjilali [BBB19] published a paper discussing the usage of a greedy insert algorithm in a setup with transportation time constraints.

2.2.3.3 Shifting Bottleneck

First described in 1998 by Joseph Adams, Egon Balas, and Daniel Zawack [ABZ88], the last constructive method type we will look at is the shifting bottleneck. In these methods, the goal is to reduce the time it takes to solve a multi-machine job-shop problem, by breaking down the global problem into smaller ones for each machine. The name shifting bottleneck stems from the main idea behind these methods. They work by picking a machine in each iteration, from the list of unscheduled machines, identified as the bottleneck machine, and solving the problem for that machine. This way, these methods are mostly used to provide a heuristic to reduce the total makespan [ABZ88].

Soon after the initial idea was proposed, other researchers published papers putting forward their implementations of the shifting bottleneck. In 1993, S. Dauzere-Peres and J.B. Lasserre [DPL93] proposed a way to use a shifting bottleneck approach to a Job-Shop Scheduling problem that had to consider delay precedence constraints (DPC), which as the name indicates are delays between the end of a job, and the start of the next one. In 1995, Egon Balas, Jan Karel Lenstra, and Alkis Vazacopoulos [BLV95], followed in 1998 by Egon Balas and Alkis Vazacopoulos [BV98], also proposed a Shifting Bottleneck implementation to deal with CDP, with the difference that the later combined Shifting Bottleneck with Branch and Bound methods.

More recently, it is common to see Shifting Bottleneck solutions combined with other heuristics to provide hybrid approaches, or with some modifications to serve in specific scenarios. Examples can be found in a paper published in 2014, by Qiao Zhang, Hervé Manier and Marie-Ange Manier [ZMM14] who proposed an adapted version of Shifting Bottleneck approach to deal with transportation constraints. Or in 2016, with two other papers, one published by B. H. Zhou and T. Peng [ZP16] who modified the Shifting Bottleneck to deal with large-scale instances, and one published by R. Mellado Silva, C. Cubillos, and D. Cabrera Paniagua [SCP16] that combined Shifting Bottleneck with Taboo search to achieve a hybrid approach.

2.2.4 Artificial Intelligence Methods

The term Artificial Intelligence first appeared in the summer of 1956 at Dartmouth College, to describe the study of conferring human-like intelligence to computational units with non-organic nature [CUB13, Cre95, ZDZ+19].

And so, in this Subsection, we will look at Artificial Intelligence approaches to the Job-Shop Scheduling problem. Methods that belong to this group rely on the available information about the system, together with intelligent algorithms, to reach suitable solutions in an acceptable time. The methods shown are further divided into Subsections of Constraints Satisfaction and Neural Networks.

2.2.4.1 Constraints Satisfaction

Used not only for the Job-Shop Scheduling problem but for many other domains, constraint satisfaction methods aims to reduce the number of potential solutions by applying a set of rules that limit the search space that requires exploration [CUB13, SM98].

In his Ph.D. thesis published in 1983, Mark Fox [FM83] described his constraints satisfaction approach to the job-shop scheduling problem, exploring some of its main challenges, such as knowledge representation, constraint selection and conflict management between constraints. Fox also proposed a knowledge system to support his solution, called Intelligent Scheduling and Information System (ISIS), which later led to the creation of other knowledge systems for scheduling. One notable example of a knowledge system that stem of ISIS is the one proposed in 1986 by Mark Fox himself, together with Stephen Smith and P.S. Ow [SFO86] called the Opportunistic Intelligent Scheduler (OPIS).

In 1995, Didier Dubois, Hélène Fargier and Henri Prade [DFP95]published their work on how to use fuzzy logic together with constraint satisfaction through a lookahead algorithm that allowed for some ability to make decisions in the presence of uncertainty. Then in 1996 the works of Erwin Pesch and Ulrich A. W. Tetzlaff [PT96] and Norman Sadeh and Mark Fox [SF96] focused on further exploring the constraint satisfaction algorithms the first by investigating how to better prune solutions in the search tree and the later by proposing a probabilistic framework to offer a heuristic for constraint selection. Further reading about constraint programming in general can be found in [Apt03, RBW06].

2.2.4.2 Neural Networks

The study of Neural Networks (NN) is an AI field that investigates how the neural systems found in living organisms can be adapted into a digital context to confer learning abilities to computational entities. Illustrated in figure 2.10) we can see the typical architecture of these networks. In this Subsection, we will look at how this ability to learn can alone or combined with other methods help solve the Job-Shop Scheduling problem [RPKKMH19].

In 1988 Yoon-Pin Simon Foo and Yoshiyaau Takejujt put forward two papers on the use of neural networks for Job-Shop Scheduling, one of them presenting "... an integer linear programming neural network based on a modified Tank and Hopfield neural network model" [FYPT88a] and the other who further advanced the first [FYPT88b]. Then in 1991 D. N. Zhou, V. Cherkassky, T. R. Baldwin and D. E. Olson [ZCBO91] also published a work on the use of neural networks to solve de JSP through Hopfield networks, describing how they overcome what they identified as a shortcoming of NN at the time, by transforming quadratic energy cost functions into linear functions. In 1994 T.M. Willems and J.E. Rooda [WR94] and in 1998 A. S. Jain and S. Meeran [JM98] further described the use of Hopfield and integer programming based solutions for the JSP.

More recently, NN mostly appears in JSP literature together with other techniques in hybrid solutions. In 2003, Yoon-Pin Simon Foo and Yoshiyasu Takejuji [MY03] used NN solutions to select dispatch rules. In 2008, Gary R. Weckman, Chandrasekhar V. Ganduri and David A. Koonce [WGK08] combined NN with Genetic Algorithms. In 2015, A. S. Xanthopoulos and D. E. Koulouriotis [XK15] used cluster analysis together with NN for the selection of dispatch rules for dynamic sequencing. In 2017 Donghai Yang and Xiaodan Zhang [YZ17] combined NN with GA to predict job due dates. And very recently in 2019, N. Rafiee Parsa, T.Keshavarz, B. Karimi and S. M. Moattar Husseini [RPKKMH19] also used a hybrid neural network approach to minimize the total makespan on only one machine.

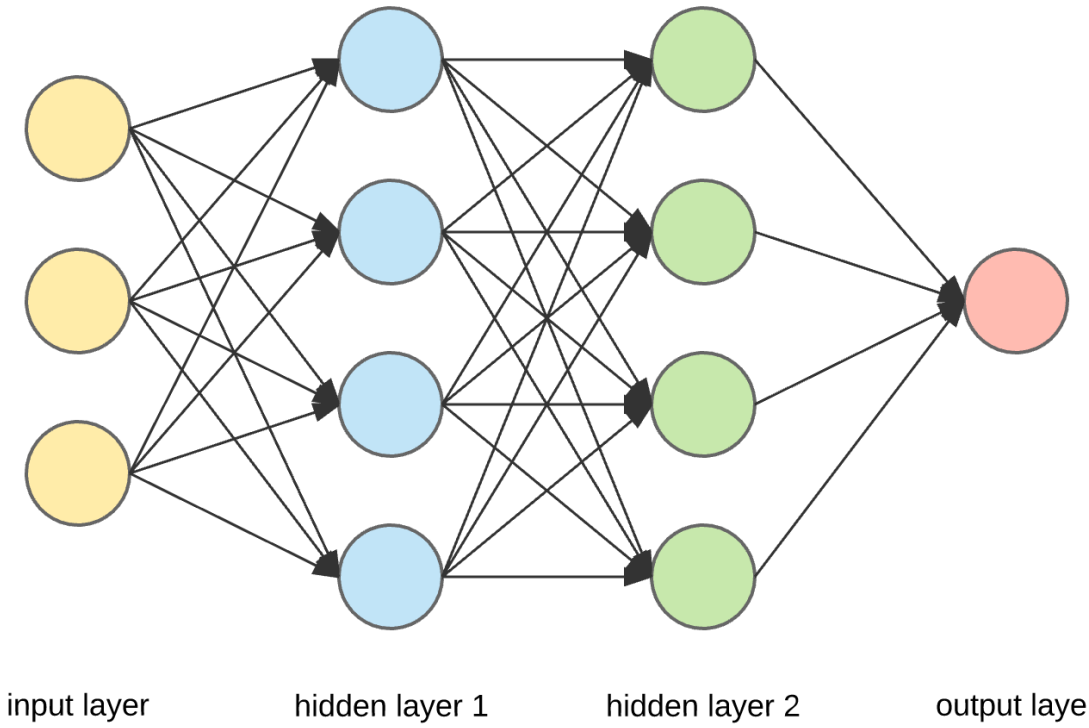

Figure 2.10: Neural Network (Source: [Sci])

2.2.5 Local Search Methods

From the idea that it is possible to reach optimal or near-optimal solutions by continuously improving an existing solution appeared local search methods. Introducing slight changes and reevaluating the result at each step, these methods believe that a guided search can find relevant solutions. Exploring only a local portion of the space state, they can provide a quick solution to problems like the JSP. The main disadvantage of these methods is being susceptible to getting trapped in a local optimum. In this Subsection, we will discuss some of the most common local search algorithms used for the JSP and how they cope with their disadvantages.

2.2.5.1 Beam Search

The first local search method we are going to look at is Beam Search (BS). Very similar to the Branch and Bound, this method only expands the most promising k nodes, whereas BnB expands all of them. For this reason, BS is faster but less accurate than BnB, making it better for problems of a decent dimension. A common variant of the BS that is used to solve the JSP is the Filtered Beam Search (FSP). FSP uses a beam width b and a filter width f. In each iteration, it selects the best f nodes with a filter function and then selects the most promising b nodes for expansion (Illustrated in figure 2.11).

In 1988, Peng Si Ow and Thomas E. Morton [OM88] published a paper describing their use of FBS for the single machine problem with tardiness and earliness and for the weighted tardiness problem, comparing their results against dispatch heuristics and neighbourhood search.

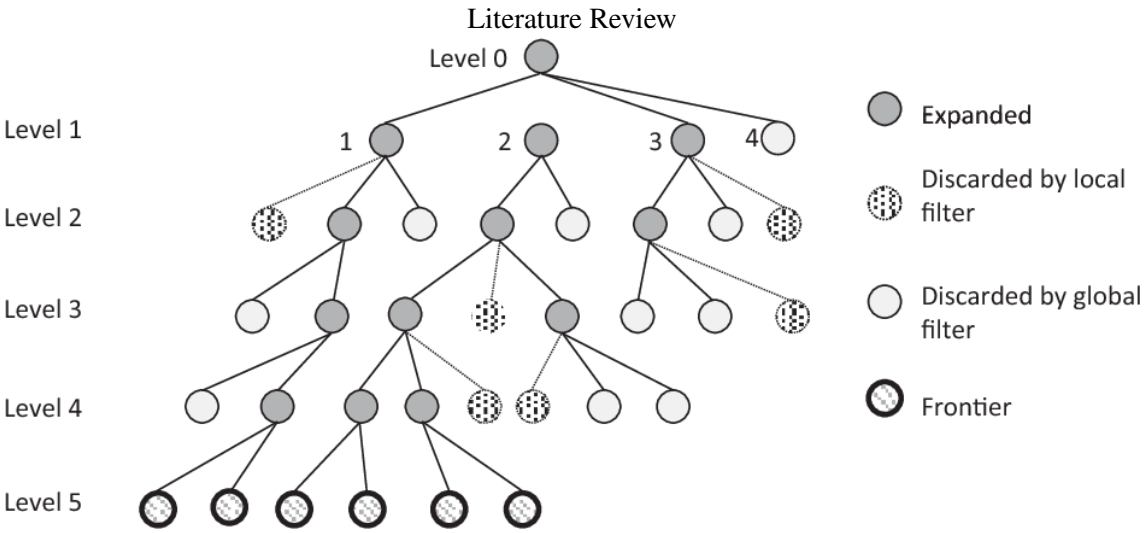

Figure 2.11: Filtered Beam Search: [MN17] (Filterwidth = 6 and Beamwidth = 3)

In 1999, I. Sabuncuoglu and M. Bayiz [SB99] also used BS to solve JSP with tardiness and makespan objectives, comparing it to dispatch rules and other search methods. Wang Shi-jin, Xi Li-feng, Zhou Bing-hai published two papers, the first in 2007 [SjLfBh07] used FBS for dynamic rescheduling describing how FBS can help cope with events like machine failure and unexpected orders and the second in 2008 [WZX08] described how they combined BS with Dispatch Rules. Later in 2010 Shijin Wang [Wan10] published a paper describing how he used FBS to deal with a JSP with transport time. More recently, in 2015, E. G. Birgin, J.E.Ferreira and D. P. Ronconi [BFR15] used BS to solve the JSP with sequence flexibility and in 2016, Mario C. Vélez-Gallego, Jairo Maya and Jairo R. Montoya-Torres citejspbs5 used BS combined with integer programming to minimize the makespan with sequence dependent setup times.

2.2.5.2 Simulated Annealing

Put forward by S. Kirkpatrick, C. D. Gelatt, Jr. and M. P. Vecchi in 1983 [KGV83] Simulated Annealing (SA) is a local search heuristic that tries to apply the mechanical process of annealing to solve optimization problems. In SA, a control variable called temperature controls the energy of the system, this temperature begins high, and gradually moves to zero, with each iteration, when the temperature hits zero the algorithm execution ends. The algorithm explores the state space, trying to converge to the best possible answer, high temperature means that the algorithm is more likely to risk moving to a worse solution to escape local optimums [vLA87].

In the same decade that SA was invented, researchers started using it to try to solve the JSP. In 1989, I.H. Osman and C.N. Potts [OP89] proposed using SA to solve the permutation flow-shop scheduling in which jobs are processed in the same order in every step of production. Then in 1992, Peter J. M. van Laarhoven, Emile H. L. Aarts and Jan Karel Lenstra [LAL92] compared their implementation of a SA solution to the JSP, achieve good results for makespan at the cost of longer running times. In 1994 Takeshi Yamada, Bruce E. Rosen, and Ryohei Nakano [YRN94]

put forward a SA implementation using Critical Block Transition Operators. In 1996, Takeshi Yamada and Ryohei Nakano [YN96] achieved positive results by combining SA with Deterministic Local Search. More recently in 2010 Rui Zhang, Cheng Wu [ZW10] proposed a SA solution to minimize the total weighted tardiness, by combining it with a bottleneck strategy. In 2011, a paper published by H. S. Mirsanei, M. Zandieh, M. J. Moayed, and M. R. Khabbazi [MZMK11] used SA for scheduling with setup times that were sequence-dependent and a paper published by Hui-Mei Wang, Fuh-Der Chou, and Ful-Chiang Wu [WCW11] used SA with parallel computing to minimize makespan. Very recently in 2018, Dayan C. Bissoli, Wagner A. S. Altoe, Geraldo R. Mauri, and Andre R. S. Amaral [BAMA18] used SA for a JSP to combined the objective of minimizing both makespan and total tardiness.

2.2.5.3 Tabu Search

The last local heuristic for optimization used in JSP that we are going to talk about is Tabu Search (TS). Similar to BS and SA, TS bases itself in the idea of exploring neighbor solutions. It mainly distinguishing itself from the others by keeping a memory of the already explored space in a list called tabu list (see algorithm 3).

Algorithm 3: Tabu Search

```
Result: Best Solution
Solution bestSolution = input.InitialSolution ;
Solution bestCandidate = bestSolution ;
TabuList = new Array[] ;
TabuList.push(bestSolution); while !stoppingConditionsSatisfied do
    Solution[] neighbors = bestCandidate.GetNeighbors(); ;
    foreach neighbor in neighbors do
        if !TabuList.contain(neighbor) && neighbor.GetFitness() >
          bestcandidate.GetFitness() then
            bestCandidate = neighbor ;
        end
    end
    if bestCandidate.GetFitness() > bestSolution.GetFitness() then
        bestSolution = bestCandidate;
    end
    TabuList.push(bestCandidate); if TabuList.size > maxSize then
        TabuList.PopFirst() ;
    end
end
return bestSolution ;
```

In 1993, Mauro Dell'Amico and Marco Trubian from Politecnico di Milano [DT93], developed a TS based solution for the JSP, achieving better results than other iterative improvement heuristics at the time. Soon after in 1994, Johann Hurink, Bernd Jurisch, and Monika Thole [HJT94] published a paper about their use of a TS heuristic to solve the flexible JSP. In 2000, Ferdinando Pezzella, and Emanuela Merelli [PM00] combined TS with Shifting Bottleneck achieve good results in similar time than other heuristics. In 2007 Mohammad Saidi-Mehrabad, Parviz

Fattahi [SMF07] implemented their TS based solution for minimizing the makespan in an environment with alternative operation sequences and sequence-dependant setups. More recently a paper from 2018, by Tadeu K. Zubaran and Marcus R. P. Ritt [ZR18] addresses the problem of parallel machines using TS and two papers from 2019, one from Renato de Matta [dM19] which used a TS heuristic to minimize the waiting time in the JSP and one from SS Dewi, Andriansyah, and Syahriza [DAS19] which was a case study using TS on the Flow-Shop Scheduling Problem (FSSP).

2.2.6 Global Search Methods

Contrary to local methods, global optimization based solutions don't base on an iterative improvement by looking at neighboring solutions. In this Chapter, we will look at two of the most popular global meta-heuristics that are currently being used in solving the JSP. The main advantage of these methods is that they are less prone to being stuck on local optimum, with the disadvantage of possibly not converging to a suitable solution. It is relevant to point out that these methods can and are often used together with local methods to overcome the shortcomings of both.

2.2.6.1 Genetic Algorithm

Proposed by John Henry Holland [Hol92], Genetic Algorithms (GA) belong to a group of Evolutionary Algorithms (EA) that are inspired by biologic behaviors. The main idea of GA is to generate a random population of candidate solutions, evaluate each one, and use nature-inspired processes to build a better generation based on the previous one. GA runs iteratively until a predetermined criterion is satisfied or there is no notable improvement. The method of knowledge transfer between generations is based on the DNA transference from a parent to a child generation, and it usually has four phases (illustrated in image 2.12). In the evaluation phase, each individual from the population is assigned a fitness value. Then in the selection phase, some individuals are selected and other discarded, with a probability related to their fitness value. The actual transference of knowledge happens in the crossover phase, where each individual from the new generation is created by using selected parts of multiple (usually two) parents. In the last phase, mutation, the newly generated individuals can suffer a random change, to ensure the algorithm doesn't get stuck in only a portion of the state space [FF95].

From the first paper to describe the use of genetic algorithms to solve the JSP, published by Lawrence Davis in 1985 [Dav85], researchers have turned to genetic algorithms to find reliable and fast solutions for various instances of the JSP. In 1991, a paper published by E. Falkenauer, and S. Bouffouix [FB91] proposes a GA based solution for the JSP with a strong emphasis on the encoding part. Runwei Cheng, Mitsuo Gen, and Yasuhiro Tsujimura published two surveys about the use of genetic algorithms. In the context of JSP, the first one in 1996 [CGT96] focusing on the representation of the problem, and the second one in 1999 [CGT99] focused on hybrid strategies. Later in 2010, Artur M. Kuczapski, Mihai V. Micea, Laurentiu A. Maniu, and Vladimir I. Cretu [KMMC10] proposed a technique that used dispatch rules to generate a near optimal initial population for the genetic approach to the JSP, this is particularly interesting because it can drastically reduce the execution time of the algorithm as shown in the paper. In 2015, Amir

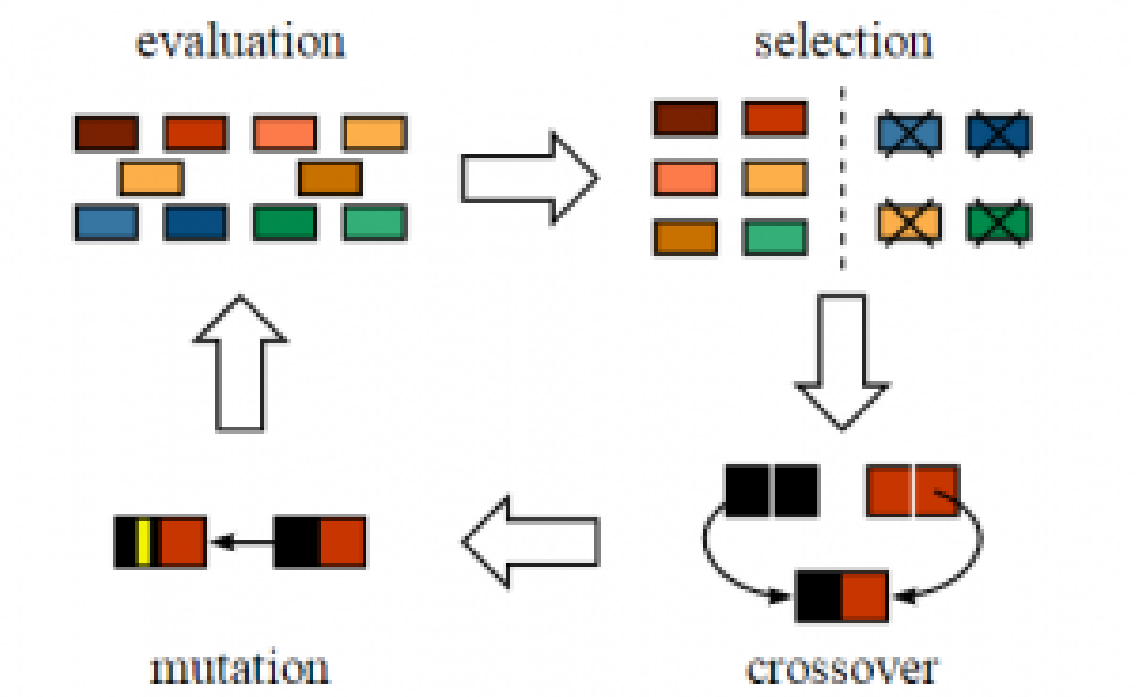

Figure 2.12: Genetic Algorithm Phases: [JAI17]

Jalilvand-Nejad, and Parviz Fattahi [JNF15] combined a GA approach with integer programming, obtaining better results than with a SA solution that served as benchmark. More recently in 2018, O. Gholami, Y. N. Sotskov, and F. Werner [GSW18] proposed a GA solution for minimizing the makespan and the mean flow time with good results comparing to other heuristics, and in 2019, X. Huang, and L.Yang [HY19] published a paper on the use of GA to solve a multi objective JSP considering transportation times.

2.2.6.2 Ant Colony Optimization

Proposed by Marco Dorigo in 1991 [CDM91], ant colony optimization (ACO) heuristics are also inspired by a phenomenon that occurs in nature. In their habitats, ants work together to find the best route from their colonies to food sources by leaving a trail of pheromones that go from one place to the other. Because pheromones evaporate, and since ants that choose the shortest path are going to leave a higher concentration of pheromones, eventually the best route will be the one with more pheromones (as illustrated in image 2.13. Similar to GA, ACO has the benefit of being a naturally parallel process, which makes it easier to benefit from more than one computational units [DMC91].

The creators of ACO hinted in a paper published in 1991 [DMC91] that among other applications, ACO could be used to solve the JSP, originating various other papers that were often combined with additional algorithms to create more robust solutions. In 2005, Christian Blum [Blu05] combined ACO with BS, followed by Kuo-Ling Huang, and Ching-Jong Liao in 2008 [HL08] who combined ACO with TS. Then in 2010, Li-Ning Xing, Ying-Wu Chen, Peng Wang,

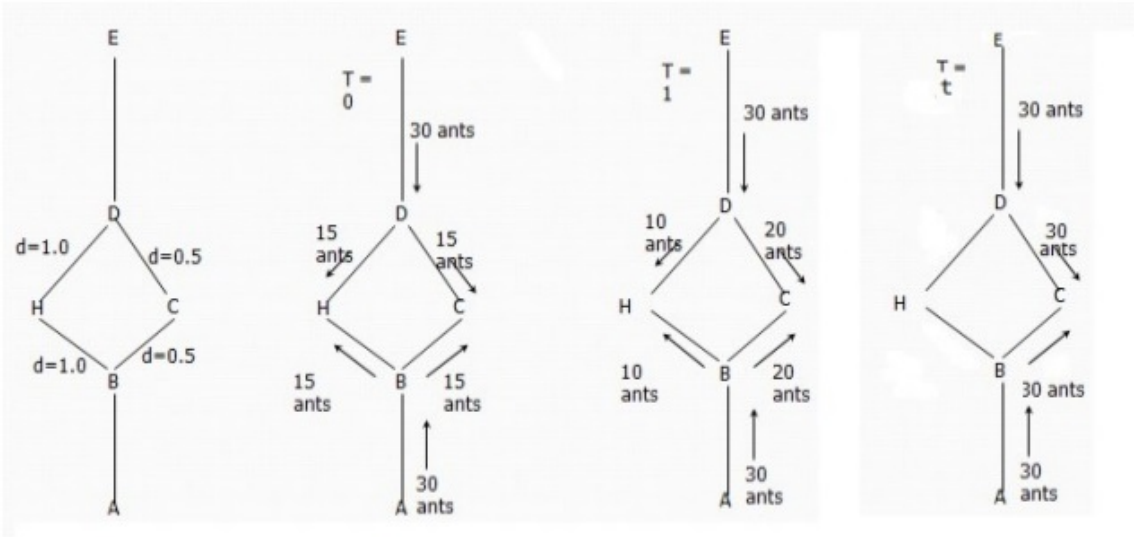

Figure 2.13: Ant Colony Optimization Example: [Das14]

Qing-Song Zhao, and Jian Xiong [XCW+10] combined ACO with a knowledge based system. In 2015, Hongquan Xue, Peng Zhang, and Shengmin Wei [XZW15] used a hybrid immunity algorithm with ACO and Boxuan Zhao, Jianmin Gao, Kun Chen, and Ke Guo [ZGCG15] combined ACO with two-generation pareto. More recently in 2017, Rong-Hwa Huang, and Tung-Han Yu [HY17] proposed an improved ACO implementation to deal with dynamic JSP.

2.3 JADE Framework

Because we've chosen the Java Agent Development Framework (JADE) platform to build our application, we've included this small Section to provide the user some knowledge about the platform's background as well as some of its main features.

2.3.1 Origin

In 1996, was established an organization called Foundation for Intelligent Physical Agents (FIPA). This nonprofit association, composed of academic and industry members, had the goal to put forward a series of rules that would standardize and regulate the knowledge about AI agents that at the time was getting low attention and lacked conformity [BCG07]. These standards, as well as some additional information about FIPA, can be found on their website [FIP05].

After FIPA put forward their standards, arose the need to validate them. With this goal in mind, in late 1998, Telecom Italia (previously known as CSELT) developed a software that later evolved to what JADE is today. Then in 2000 Telecom Italia started distributing JADE as an open software under the LGPL allowing for developers all over the world to use the software to build their agent-based applications [BCG07].

2.3.2 Features

Today JADE is a FIPA compliant, Java framework for developing multi-agent systems. Following FIPA specifications, JADE includes the following features [BPR]:

- An Agent Management System (AMS), which manages access to the agents in the platform, any required authentication/authorization rules, as well as the registry of agents.
- A Directory Facilitator (DF) that works as a yellow pages service for the agents.
- An Agent Communication Channel (ACC) which ensures reliability and accuracy in the communication between agents inside or outside the platform.
- An architecture that allows the multi-agent environment to exist in multiple computational units with a single instance of the software in each host.
- The thread managing logic to enable each agent to run on its thread without additional effort.
- A GUI that allows for simple management and efficient monitoring of the agents and their containers.
- A transport abstraction to efficiently exchange messages between agents, automatically selecting the fastest way to deliver a message based on the distance between the two agents (for instance in the same host messages can be sent as Java objective while in different hosts they cannot).
- An agent naming system that ensures each agent is identified uniquely in the framework through agent identifiers (AID).
- A logging system.

Furthermore, the JADE framework also makes available a series of agent behavior templates to represent common agents' actions or interactions (illustrated in image 2.14). More information about JADE and the JADE Framework are available at their website [JAD19a, JAD19b].

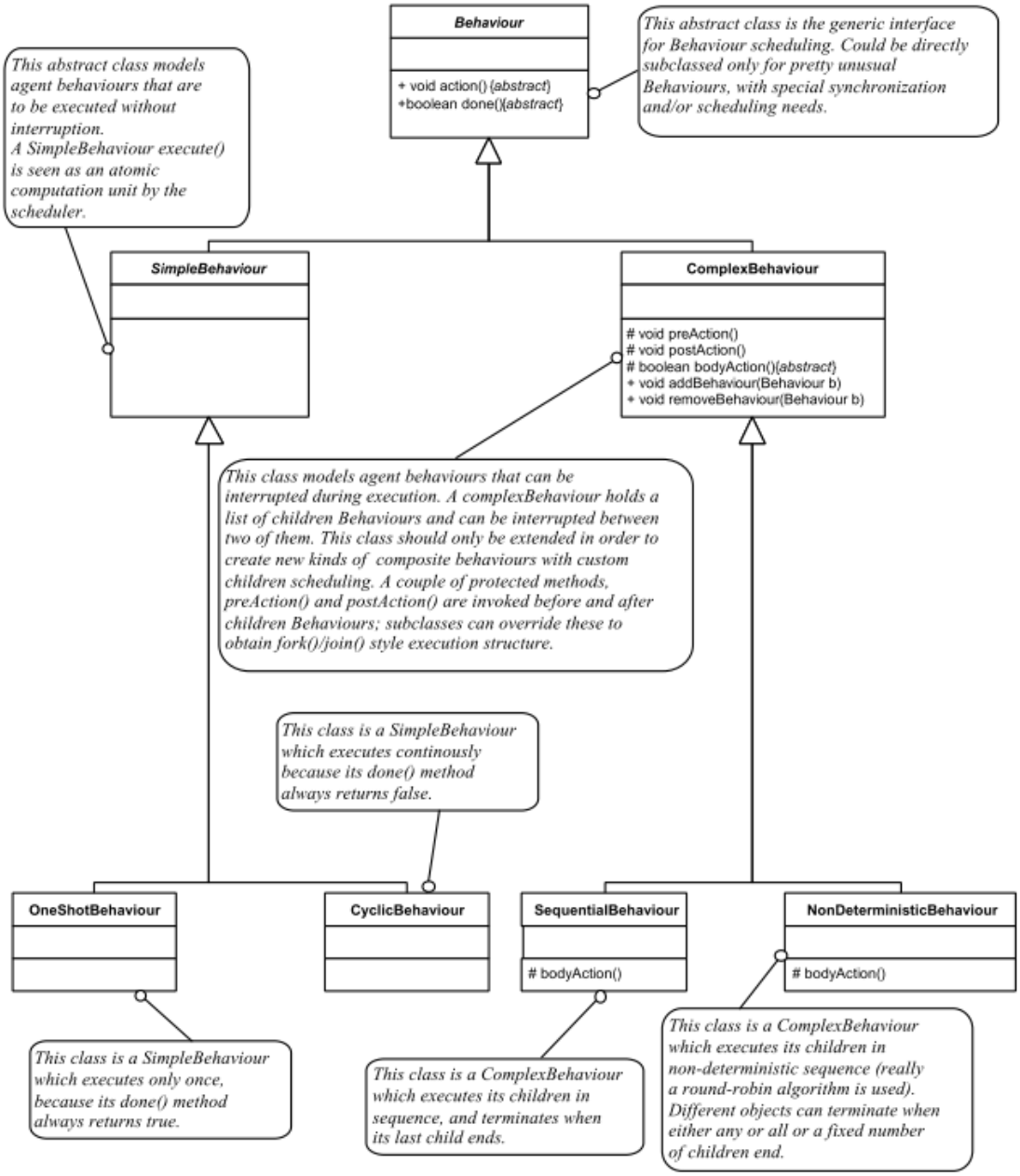

Figure 2.14: Jade Behaviours: [BPR]

Chapter 3

Problem Description

In this Chapter, we will look at the Job-Shop Scheduling Problem definition. To do so, we have divided the information into two different Subsections. In the first subsection (3.1), we present the problem and show all the different variables that may come into play. In the second Subsection (3.2), we will describe how we formalized the problem that our implementation presented in Chapter 4 intends to solve.

3.1 Job-Shop Scheduling Problem

When we talk about the Job-Shop Scheduling Problem (JSSP), in reality, we are talking about a whole myriad of problems with different variables and rules. To fully define this universe, we will first look at one of the most simplistic JSP, and from there, we will gradually add complexity to ensure a smooth flow of information.

3.1.1 Single Machine

A simplistic instance of the JSP is when we have only one machine and a set of jobs, and we must assign the order in which the jobs are going to be processed. In the most simple of cases, we only know how much time each job takes to process and its due date (illustrated in table 3.1). Because in this formulation the total makespan is constant, and there are no setup times, usually the goal is to either minimize the tardiness (which is given by Max(0, CompletionDate- DueDate)), to minimize the earliness (which is given by Max(0, DueDate-CompletionDate)), or a combination of both.

Job	Processing Time	Due Date
A	3	4
B	4	5
C	5	6

Table 3.1: Job Information Example

But even for such a seemingly simple problem, there isn't an efficient rule to consistently solve these problems. Some heuristics, as seen in Chapter 2, are the earliest due date (EDD) and the remaining amount of work can provide optimal or near-optimal solutions, but the number of possible states (example illustrated in table 3.2) for this problem is still $n!$ with n being the number of jobs.

Sequence	Total Tardiness	Total Earliness	Sum
A B C	8	1	9
A C B	9	1	10
B A C	9	1	10
B C A	11	1	12
C A B	11	1	12
C B A	12	1	13

Table 3.2: Possible States

3.1.2 Priorities and Setup Times

Priorities and setup times are two other dimensions that can add complexity to the JSP. In the real world, machines can require some time between the end of a job and the start of the next one. This setup time can be the same for every two jobs, but in most real-life scenarios, it is possible to define a setup-matrix in each machine. Priority is also a crucial aspect of closing the bridge between theory and the real world. Since not every production order as the same value for the manufacturing facility, it is a common practice to assign different priorities. After adding these two new variables to the problem, we also have to rethink what goals we can establish. By adding the setup times, we may now want to minimize the total time spent on it, because in a factory this time usually comes with an associated cost. With the institution of a priority system, it may also be relevant to ask the weighted tardiness and earliness. In table 3.3 we can see an updated version of how a job information table will look like with these changes. And in table 3.4 we can see an example of how a setup-matrix table looks like.

Job	Processing Time	Due Date	Setup Time	Priority
A	3	4	1	1
B	4	5	2	2
C	5	6	3	3

Table 3.3: Job Information Example Updated With Priority and Setup Times

From/To	A	B	C
A	0	1	2
B	3	0	4
C	5	6	0

Table 3.4: Example of a Setup Matrix

The solution space of the problem described for the example input is illustrated in table 3.5 with setup times only, and in table 3.6 considering the priority.

Sequence	Total Tardiness	Total Earliness	Sum	Total Setup Time
A B C	8	1	9	5
A C B	9	1	10	8
B A C	9	1	10	5
B C A	11	1	12	9
C A B	11	1	12	6
C B A	12	1	13	9

Table 3.5: Possible States After Introducing Setup Times

Sequence	Weighted Tardiness	Weighted Earliness	Weighted Sum	Total Setup Time
A B C	11	0.5	11.5	5
A C B	10	0.5	10.5	8
B A C	10.5	1	11.5	5
B C A	8.5	1	9.5	9
C A B	9	1.5	10.5	6
C B A	8	1.5	9.5	9

Table 3.6: Possible States After Introducing Setup Times and Priorities

One of the first things that come to mind, when looking at the updated tables, is that the solution space size did not increase in comparison to the previous example. Although this is true, one can observe that sometimes the different goals are conflicting, which is consistent with real-life objectives that factories may have. This increase in complexity also adds an additional logic that solving strategies must incorporate to reach desirable solutions.

3.1.3 Maintenance and Non-Working Times

Up until this point we've considered that our hypothetical machine is always available to process our jobs, but in the real-world, machines have a working and non-working period and require regular maintenance. If we consider that, we can quickly find out that different allocations of our jobs may result in different completion times of the last job (makespan). As illustrated in figure 3.1, by changing how we choose to occupy or available time on the machine, we can have a different amount of time between jobs due to the time the machine is paused (paused time), resulting in the increased makespan. In some real-life scenarios, paused times can be quite expensive. A notable example of this is industrial ovens, which due to the high costs of starting them are rarely turned off, so if these ovens aren't processing a job the factory is losing money. And of course, a smaller makespan is desirable since it increases the amount of production the factory can get done in a certain timeframe.

A common question that may arise is why there is a distinction between maintenance times and regular non-working times. Besides being conceptually different, non-working times are generally more static than maintenance operations. A working period usually stays the same, and while some maintenances are regular, it is not uncommon for a machine to require unscheduled maintenance.

In Chapter 2 we've mentioned a work that used an insertion technique that only after placing the jobs scheduled maintenance operations. Besides that, jobs and machines can have special ways to interact with these situations. Some jobs can spread across working time, meaning they can start on working time then paused and resumed in the next working time. There are also some machines that typically don't require supervision that although having to be started in working time may continue to process and even finish in non-working time.

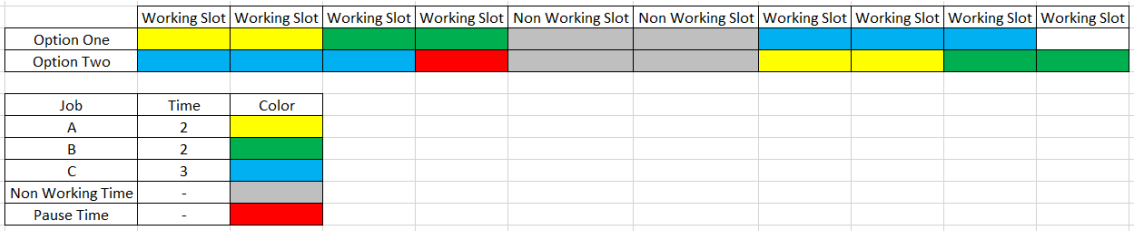

Figure 3.1: Non-Working Times

3.1.4 Capacity and Quantity

In our previous examples, jobs are a somewhat abstract concept of a task that requires a machine to complete it. In the real-world these jobs usually are associated with physical materials that have a shape and volume. And so, another big aspect of manufacturing is machine capacity. Because different machines have different capacities, there isn't always a match between the size of the job and the capacity of the machine. For this reason some jobs may have to be divided into two smaller jobs if the job doesn't fit the machine (batch production). On the other hand, it is also possible that different jobs may be processed by only one machine at the same time, promoting the utilization of that machine. To manage this, it is not unusual that a job has an associated type and a quantity that will then map to a capacity unit (illustrated in tables 3.7 and 3.9).

Capacity Type	Size	Product Type
Small	1	P1
Large	3	P1

Table 3.7: Capacity Table

It is also notable that in these environments, it is also common for processing times to have a fixed component and a component that varies with the quantity (illustrated in tables 3.8 and 3.9).

Product Type	Fixed Processing Time	Varying Processing Time
P1	5	1
P2	10	2
P3	10	3

Table 3.8: Processing Times Table

Although this may not seem relevant in a one machine context, in a real-life context with multiple machines and a great variety of products and capacity classes these two variables can

greatly increase the complexity of the problem. It is also not uncommon for manufacturers to require that some machines are at least a percentage full before starting. In some even more complex cases, it is even possible to have tables that define which jobs can be combined and the processing time in the case they do.

Job	Capacity Type	Product Type	Quantity	Processing Time	Size in Machine A
A	Small	P1	10	15	30
B	Small	P1	10	15	30
C	Large	P2	5	20	15
D	Large	P2	5	20	15
E	Large	P3	5	25	15

Table 3.9: Typical job representation regarding capacity and varying processing time

3.1.5 Job Precedence and Products

By moving further down the path of making our problem definition more suitable for a real-life scenario, we will now talk about job precedence. In a factory, we must complete some jobs before allocating others. If we think of jobs as steps in the production flow of a product, we can quickly understand that they are usually in groups representing all the steps from raw material to a finished product. Having this in mind, real-life representation of the problem would look more like the one in tables 3.10 and 3.11.

Product Type	Due Date	Priority	Quantity	Capacity Type
P1	5	1	10	Big
P2	10	2	20	Small

Table 3.10: Example list of products

Job	Product	Step	Order
J1	P1	Cooking	1
J2	P1	Welding	2
J3	P1	Forging	3
J4	P2	Welding	1
J5	P2	Forging	2

Table 3.11: Example list of jobs belonging to products

But on the other hand, this new information makes us have to rethink our goals and representation. The first thing we may notice is that the due date, the quantity and the priority now belong to the product. By having our jobs grouped by a product, we now have to consider for how long will the material we are working on occupy space in the factory. The amount of time from the start of the first job and the end of the last job is called makespan and is yet another variable we can minimize (not to mistake with the total makespan of the production schedule). It also notable that now machines can store the time to process a job by product or by job (and in some cases, the product group). Furthermore, in some special cases, jobs of different products can have precedence. And

some jobs within the same product can be performed in parallel, or be divided into two jobs with smaller quantities to fit a machine at a certain step (batch production).

3.1.6 Multiple Machines

After exploring the concepts of jobs and products, we will finally talk about the existence of multiple machines. Contrary to the other variables we've discussed, the presence of more than one machine is less about adding complexity and more about increasing the size of the search space. By having more than one machine, each time we select a job to allocate, we also need to specify where is the job going to be performed. If we look at table 3.12, we can see that with just two jobs, by adding a second machine we now have eight options. And if we add one more machine, we get eighteen options (illustrated in table 3.13). Note that this formula $n! * m^n$ takes into consideration options that do not use all machines, a way to reduce the space is to discard these scenarios. We will add an example but by using the formula $n! * m^n$ we quickly realize that even a small scenario of $n = 5$ and $m = 5$ will have 375000 options.

Sequence	Machine Selection
AB	A1B1
AB	A1B2
AB	A2B1
AB	A2B2
BA	B1A1
BA	B1A2
BA	B2A1
BA	B2A2

Table 3.12: Two jobs and Two machines

Luckily for us, in real-life scenarios machines are rarely able to do every job. The instance of JSP where every machine can do every job is a known relaxation of the problem called Flexible Job-Shop Scheduling Problem (FJSP). There is also a special case of the JSP where jobs are grouped into products, each product has one job in every machine, and the product order is always the same, called Permutation Flow Shop Scheduling Problem (PFJSP). Back to the real-world, each machine can perform only certain types of jobs. To better represent this reality, it is common to associate a list of services to each machine and to specify for each job what is the service it requires. In table 3.14 we can see our product orders, then in table 3.15 we specify the jobs it requires and in table 3.16 we have an example of a service table.

3.1.7 Multiple Resources

Although we do not address this particular dimension of the JSP in our implementation, we believe it would still be relevant to include a Subsection about multiple resources. In a factory sometimes may be relevant to schedule not only to allocate jobs to machines but to other resources such as tools or employees. Employees have vacations and may not always be available when machines are. Furthermore, some machines may require a special skill or certification to be operated by an

Problem Description

Sequence	Machine Selection
AB	A1B1
AB	A1B2
AB	A1B3
AB	A2B1
AB	A2B2
AB	A2B3
AB	A3B1
AB	A3B2
AB	A3B3
BA	B1A1
BA	B1A2
BA	B1A3
BA	B2A1
BA	B2A2
BA	B2A3
BA	B3A1
BA	B3A2
BA	B3A3

Table 3.13: Two jobs and Three machines

Product Type	Due Date	Priority
P1	10	3
P2	15	2
P3	20	1

Table 3.14: Example Product List

Job	Product	Step	ServiceId	Order
J1	P1	Cooking	1	1
J2	P1	Welding	2	2
J3	P1	Forging	3	3
J4	P2	Cooking	1	1
J5	P2	Welding	2	2
J6	P2	Forging	3	3
J7	P3	Welding	2	1
J8	P3	Forging	3	2

Table 3.15: Example Job List with Services

Machine	Name	Service Provided	Service Id
M1	Cooking and Welding Machine	Cooking	1
-	-	Welding	2
M2	Cooking and Forging Machine	Cooking	1
-	-	Forging	3

Table 3.16: Example Machines Service List

employee. On the other hand, tools are more flexible as we only need to track how many tools exist and how many of those are being used.

Machine	Name	Service Id	Certification Id
M1	Machine 1	1	1
-	-	2	2
M2	Machine 2	1	1
-	-	3	3

Table 3.17: Example Machines with Required Certification

Employee	Name	Certification Id	Certification Id
E1	Employee 1	1	1
-	-	2	2
E2	Employee 2	1	1
-	-	3	3

Table 3.18: Example Employees with Certification

Day	Employee
20/01/2020	E1
21/01/2020	E1
20/01/2020	E2
21/01/2020	E2

Table 3.19: Example Employees Vacations

Tool	Name	Quantity
T1	Hammer	50
T2	Mold	50

Table 3.20: Example Tools

Job	Service	Tool	Tool Quantity
A	1	T1	5
B	2	T2	5

Table 3.21: Example Jobs with Tools Requirement

After adding these variables the formulation of the problem may look like something along the lines of what is illustrated in tables 3.17, 3.18, 3.19, 3.20, and 3.21.

3.2 Addressed Problem

In the next Chapter (4), we will explain how we have implemented our solution to JSP. However, before we start explaining our solution, as there are many ways to interpret the JSP, there is still

the matter of explaining what exactly is the problem that we are going to solve. And so we have included this small Section to clarify how much of the presented dimensions we are going to be addressing.

In our implementation, we will deal with multiple jobs and multiple machines. Jobs are organized in sequences that represent the flow from an input material to a fully assembled product. These products have the logic of due dates, priorities, and capacity. Then each job inside the product flow has its ordering, required service, and instructions on how to interact with non-working times. Machines will have a default setup time, a matrix to further detail setups times between certain services, a table to calculate the processing time of each job, a table for how to handle capacities, a list of their working-times, and the production type (e.g. batch production). Features that are not included, are the differentiation between non-working times and maintenance, employees, and tools. Finally, the only goals we will be addressing are the total setup type, the average tardiness, and the makespan.

Chapter 4

Implementation

In this Chapter, we will describe how we have implemented a solution to the problem described in the previous Chapter 3.

4.1 Process initiation

When our program starts, the first thing it does is initialize a JADE instance that will manage our agents, and then sets up and HTTP Server to begin listening for requests.

The process is initiated by an HTTP request. Currently the server will respond with one of the values specified in table 4.1 to a POST request with the parameters specified in table 4.2.

Response Code	**Message**	**Meaning**
200	Already Running	There is another scheduling in progress (The program currently can only run one instance at a time).
200	This route only accepts POST requests	The verb was incorrect.
200	Scheduling started successfully	The request was understood and the scheduling process has started.
400	The server could not understand your request	The server could not understand the request. This is triggered by a fail in the JSON parsing of the request.

Table 4.1: Program HTTP responses

If the request is correctly formatted and is understood by the server, the server will then instantiate a request handler that will load all the necessary data from the database, starting the next phase of the program.

Parameter	Description
IsRescheduling	Our program allows not only for a regular rescheduling but also to reschedule in response to machine failure. If we set this value to "true" then the machine specified in the parameter ResourceId will be considered to have failed.
Resource Id	This parameter specifies the id of a specific machine. If a valid resource id is passed to the application and IsRescheduling is set to "true", it means that the specified machine has suffered from failure and can no longer be considered for the scheduling problem. Furthermore, any job scheduled in this machine will need to be rescheduled. If IsRescheduling is set to "false", this value will only be used to get the scheduling plan associated to this machine, which will then allow us to only get other machines and relevant jobs within that scheduling plan.
Job Lookup Limit	In the system we implemented our scheduler, the jobs have already been created and have suggested starting dates. This value specifies what is the maximum date at which jobs stop being considered for the current scheduling round. Furthermore, this date also defines the lookup limit for immutable jobs. These jobs are the ones who have already started, and in the case of partial scheduling they are loaded to calculate machine availability and transitional setup times.
Scheduling Start	This date specifies the earliest date at which jobs may be allocated. This value should be close to the current date but with some margin to prevent the schedule being obsolete right after being released.
Scheduling End	This date specifies the latest date at which jobs may be allocated. If a given solution cannot allocate all jobs, it will output a warning saying that a certain job could not be allocated (resulting in a schedule with unscheduled jobs).
Reschedule All	This variable controls the scope of the scheduling. If we are scheduling in response to a machine failure, setting this variable to false will make it only consider jobs from that machine for scheduling. (Improving running time but reducing the quality of the scheduling)
Due Date Criteria	This variable details the relative weight of the tardiness goal.
Makespan Criteria	This variable details the relative weight of the makespan goal.
Setup Time Criteria	This variable details the relative weight of the setup time goal.
Mode	Our program allows three different modes. These modes can be "fast", "dispatch", and "genetic". Further in this Chapter we will detail each of these modes.

Table 4.2: Program parameters passed by HTTP

4.1.1 Data Loading

The data we use comes from a database attached to an MES instance provided by Critical Manufacturing. To only get the required information from the database, we broke the process of retrieving data into two phases. In the first phase, we get the master data from our materials (the name given to the materials that are following the process of becoming a product), and our resources (machines). Then in the second, we already have our scope defined, so we load the related data (processing times, setup times, etc). This data stays stored in memory during the scheduling process and it's released after saving the job allocation back to the MES database.

4.1.2 Master Data Loading

In our master data loading phase, we load our materials and resources according to the scope defined by our request. The first thing we do is use the "ResourceId" parameter to determine what is the scheduling plan we are considering. Then, we load the master data from every machine, that provides a service required, by at least one of the jobs in a material being considered. If we are dealing with machine failure, we do not consider the specified resource. We then load the materials that we are going to reschedule. If we are doing full scheduling, we load every material with jobs up until the job lookup limit. But if we are dealing with machine failure we either load every material (if reschedule all is true), or we only load the materials with jobs in the specified machine (if reschedule all is false). Materials that are not being considered for rescheduling but are in the specified time frame are have only their jobs (not the material data) loaded as immutable jobs.

4.1.3 Related Data Loading

After we have loaded our materials and resources, we pass them to a generic data loader to get all the additional required information. This way we can easily extend our scheduler do deal with additional disruption types if we have to. In the case of the materials, we only have to load its jobs (remember a material has a set of jobs it has to complete to become a final product). And in the case of resources, we need to load the list of provided services, the setup matrix, the processing times, the capacity values, and the working times.

4.1.4 Summary and Structure of the Loaded Data

In this step we have loaded all the necessary data to begin and complete our scheduling problem. The only moment when we will need to interact with the database again, is when we have to save our allocated jobs.

To summarize, after loading our data, we now have a list of materials and a list of resources, and we are ready to move to the next phase. To better help the reader understand what was loaded we've created a series of table that highlight the relevant attributes of each of the business classes involved. In table 4.3 we have the resource, in table 4.4 we have the material and in table 4.5 the job. Then in table 4.6 we have the setup matrix transition, in table 4.7 the working interval, in table 4.8 the processing time and in table 4.9 the capacity.

Field	Description
Resource Id	An id that uniquely identifies the resource.
Name	The name of the resource.
Default Setup Time	The setup time to be used if there is no setup time specified in the setup matrix. Only applies if we are transitioning from a service to a different one.
Schedule Job Type	If set to 1 it means this job does not require a resource.
Can Process in Non Working Time	If set to true this resource can process and finish in Non Working Time. Still has to start on working time.
Provided Services	A list of the service ids provided by this resource.
Setup Matrix Transitions	A list of setup matrix transitions. The setup matrix definition is defined in it's own table.
Processing Times	A list of processing times. The processing time definition is defined in it's own table.
Capacity Values	A list of capacity values. The capacity value definition is defined in it's own table.
Working Intervals	A list of working intervals managed by the resource throughout the scheduling process. The working interval definition is defined in it's own table.
Scheduled Jobs	A list of scheduled jobs that do not require scheduling. The job definition is defined in it's own table.
Unscheduled Jobs	A list of unscheduled jobs that require scheduling. The job definition is defined in it's own table.

Table 4.3: Structure of a Resource

Field	Description
Material Id	An id that uniquely identifies the material.
Name	The name of the material.
Priority	The material priority.
Hot	A special priority that precedes regular priority.
Due Date	The date at which the material must be ready. After this point the material starts to accumulate tardiness.
ProductName	The name of the product that will result from finishing the material's manufacturing process.
Product Group Name	The name of the product group. (products with similar properties may share this value.
Capacity Class	The capacity class used to resolve machine capacity tables.
Jobs	A list of jobs. The job definition is defined in it's own table.

Table 4.4: Structure of a Material

Field	Description
Job Id	An id that uniquely identifies the job.
Service Id	The id of the service required by the job.
Service Name	The name of the service required by the job.
Schedule Job Type	If set to 1 it means this job does not require a resource.
Can Spread Across Working Times	If set to "true" it means a job can be interrupted in a working time and resumed in the next.
Scheduled Quantity	The amount of units to be produced.
Flow Path	The order in the production flow.
Planned Quantity	To be determined by the scheduler. The amount of units this jobs will process (if we are in batch production this value may be a portion of the Scheduled Quantity).
Planned Process Time	To be determined by the scheduler. The amount of time it will take for the job to be processed.
Planned Start Date	To be determined by the scheduler. The date the job is planned to start.
Planned End Date	To be determined by the scheduler. The date the job is planned to end.
Setup Time	To be determined by the scheduler. The amount of setup time the job will require.

Table 4.5: Structure of a Job

Field	Description
From Service Id	Service Id from the service that it is transitioning from.
To Service Id	Service Id from the service that it is transitioning to.
Value	Time value for the specified transition.

Table 4.6: Structure of a Setup Matrix Transition

Field	Description
Starting Date	Start date of the Working Interval.
Ending Date	End date of the Working Interval.

Table 4.7: Structure of a Working Interval

Field	Description
Key Type	Specifies what is the value of the material/job to compare to the key value. "A" for Product name, "G" for Product Group, "S" for Service, "D" for Default, in that priority if there are multiple entrances.
Key Value	Value of the specified key. Used for lookup.
Fixed Time Value	Fixed Time for the specified entry.
Variable Time Value	Variable time for the specified entry. Depends on the quantity.

Table 4.8: Structure of a Processing Time Entry

Implementation

Field	Description
Capacity Class	Capacity Class Key.
Value	Value representing how many units of that capacity class that resource can processed at the same time.

Table 4.9: Capacity Class Entry

4.2 Scheduling Manager

The scheduling manager is the entity that receives the data from the data loader and continues the process triggered by the initial request. It is responsible for agent management, the evaluation, and tracking of solutions, and the generation of scheduling priorities between materials.

Our implementation defines two different agents. The first type of agent is called Material Agent. After receiving the data from the data loader, the manager immediately instantiates one Material Agent per material. Later in this Chapter, we will explore how these agents work, but at this point, after being instantiated, they will idle waiting to receive a message. The other type of agent is called Solution Agent. A Solution Agent receives a Scenario (list of materials and resources), the starting time and an ordered list of material IDs that specify in which order he should schedule the materials in the scenario. In this Section, we will focus on the three strategies that the manager may use to determine the number of Solution Agents it will instantiate, and how it decides what material order to pass.

4.2.1 Fast and Dispatch Scheduling

Because the Fast and Dispatch configurations are simple, we have grouped them in this Subsection. If the manager received one of these two options, the first thing it does is order the materials, first by priority, then by the due date. If the option is Fast, a single Solution Agent is instantiated and receives this material order. If the option is Dispatch, then it instantiates some Solution Agents corresponding to the number of materials, in a way that each material is the first one exactly once (illustrated in table 4.10 for five jobs).

Combination	Material Sequence
Original	A B C D E
1	A B C D E
2	B A C D E
3	C A B D E
4	D A B C E
5	E A B C D

Table 4.10: Dispatch Ordering example with 5 Jobs

4.2.2 Genetic Scheduling

The genetic configuration is further complex than the other two. When the manager is first instantiated, it generates a population of 30 initial orderings. These initial orderings are generated using

a strategy that combines randomness and a heuristic that launches them in a desirable direction. For each ordering that we generate the process is the following:

- The first step is to shuffle the materials that come from the database with random number generation.
- Then we pick the first element in the resulting shuffled list.
- From here we pick the closest material to the selected material, recursively until there are no materials left.

Note that because the first material is randomly selected, and because our distance function can frequently cause ties, the orderings will be different from each other.

This distance is based on the estimated setup time of having each material in front of the selected material. If the estimated setup time is the same, the untie is made by looking at the due time, and priority of the next material. To estimate this setup time, what we do is look at the definition of the products they are representing, and sum the average transition time for each two services they require and can be performed by the same machine. To speed up this process, if the mode is genetic the database loader will have pre-calculated, in the loading phase, a table representing these transitions, as illustrated in table 4.11.

From/To	**A**	**B**	**C**
A	0	1	2
B	3	0	4
C	5	6	0

Table 4.11: Table of estimated setup time on product transition

This process is only used to generate the first generation of sequences. Currently our program, under the genetic configuration, runs five generations. With each generation besides the first being obtained using a standard genetic algorithm implementation.

Each individual ordering that is generated and passed to a Solution Agent will result in a solution (in a processed we will later explore). That generated solution will have it's own value for tardiness, makespan and setup, which we will use to assign a fitness to that ordering.

To calculate the fitness of each individual we first calculate the maximum tardiness, makespan, and setup time. From there the fitness is given by the formula $1/(Tardiness/MaxTardiness * TardinessFactor + Makespan/MaxMakespan * MakespanFactor + SetupTime/MaxSetupTime * SetupTimeFactor)$ (since this may cause division by zero if the makespan factor is 0, in this case it returns 1000). This process results in a table that will be used for the selection phase (illustrated in table 4.12 with only 8 entries to make it smaller).

The values in the fitness table will then be combined with random number generation to select 28 materials orderings for crossover. An individual with higher fitness has a higher chance of being selected. The two top material ordering automatically occupy two slots, so the crossover phase will only generate 28 new individuals (elitism).

The crossover uses 2 randomly picked crossover points and applies shifting crossover to obtain two child orderings from the two parents (illustrated in image 4.1). After the crossover we apply a shifting mutation with a chance of 1 in 200 (illustrated in image 4.2)

Material Ordering	Fitness	Minimum Bound	Maximum Bound
CADBE	3	0	3
DACEB	5	3	8
ACDBE	8	8	16
ABCDE	2	16	22
EDCBA	6	22	28
ADCBA	8	28	32
BCDAE	2	32	34
DAEDB	10	34	44

Table 4.12: Example Fitness Table

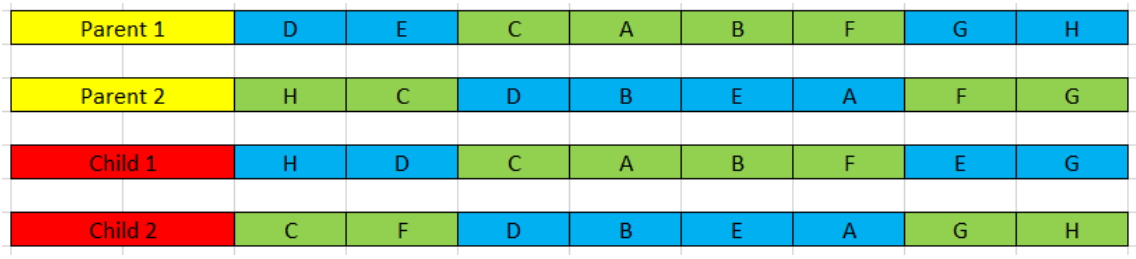

Figure 4.1: Shifting Crossover

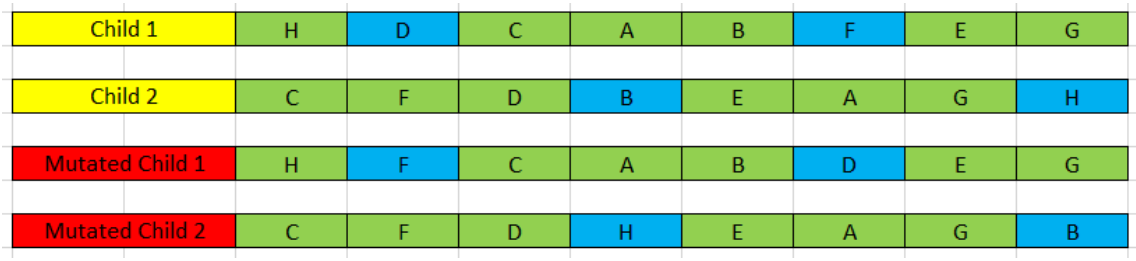

Figure 4.2: Shifting Mutation

4.2.3 Launching and Collecting solutions

After determining the desired set of orderings it wants to test, the manager will instantiate a Solution Agent for each ordering. From here the manager job is to wait for the Solution Agents to communicate their solutions. The genetic mode as mentioned ends after 5 generations while the others wait as soon as every AgentSolution has communicated its solution. The manager keeps track of the best solution, and when the scheduling is completed it terminates every agent, saves the best solution and changes the listener mode back to accepting requests (illustated in figure 4.3.

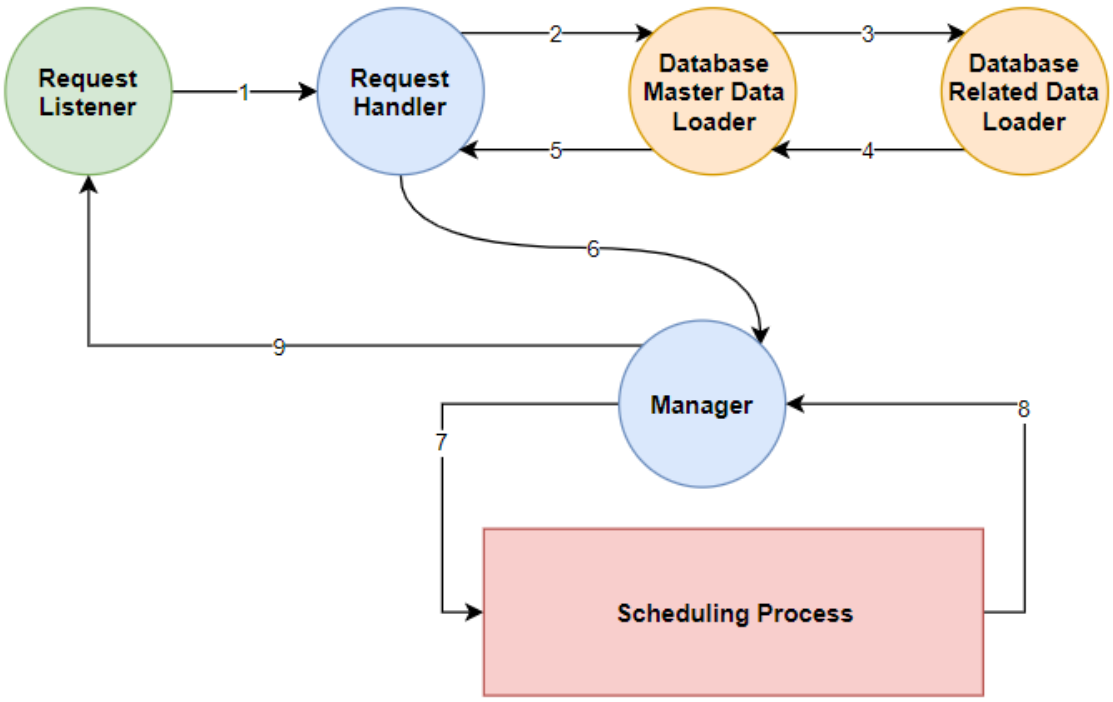

Figure 4.3: Program Flow

4.3 Agent Behaviour

In this Section we will explain the behaviour of our agents, as well as the interaction between them.

4.3.1 Solution Agent

The solution agent is initialized with a clean Scenario (a list of machines and materials without any allocations) and the material scheduling priority order that was assigned to it by the Manager in the previous step.

This agent is responsible for three things:

- Contact the first Material Agent in the list, and provide it with the scenario and the ordering so it can schedule its jobs and pass it to the next agent in the list.
- Receive the scenario with all the allocated jobs from the last Material Agent in the list, and communicate it to the Manager. For this effect, this agent signs its messages with its AID in the reply-to field of the FIPA compliant messages.
- Count the time from the moment it sends the starting message to when it receives the completed scenario, to provide the necessary control over the process.

This process in illustrated in figure 4.4.

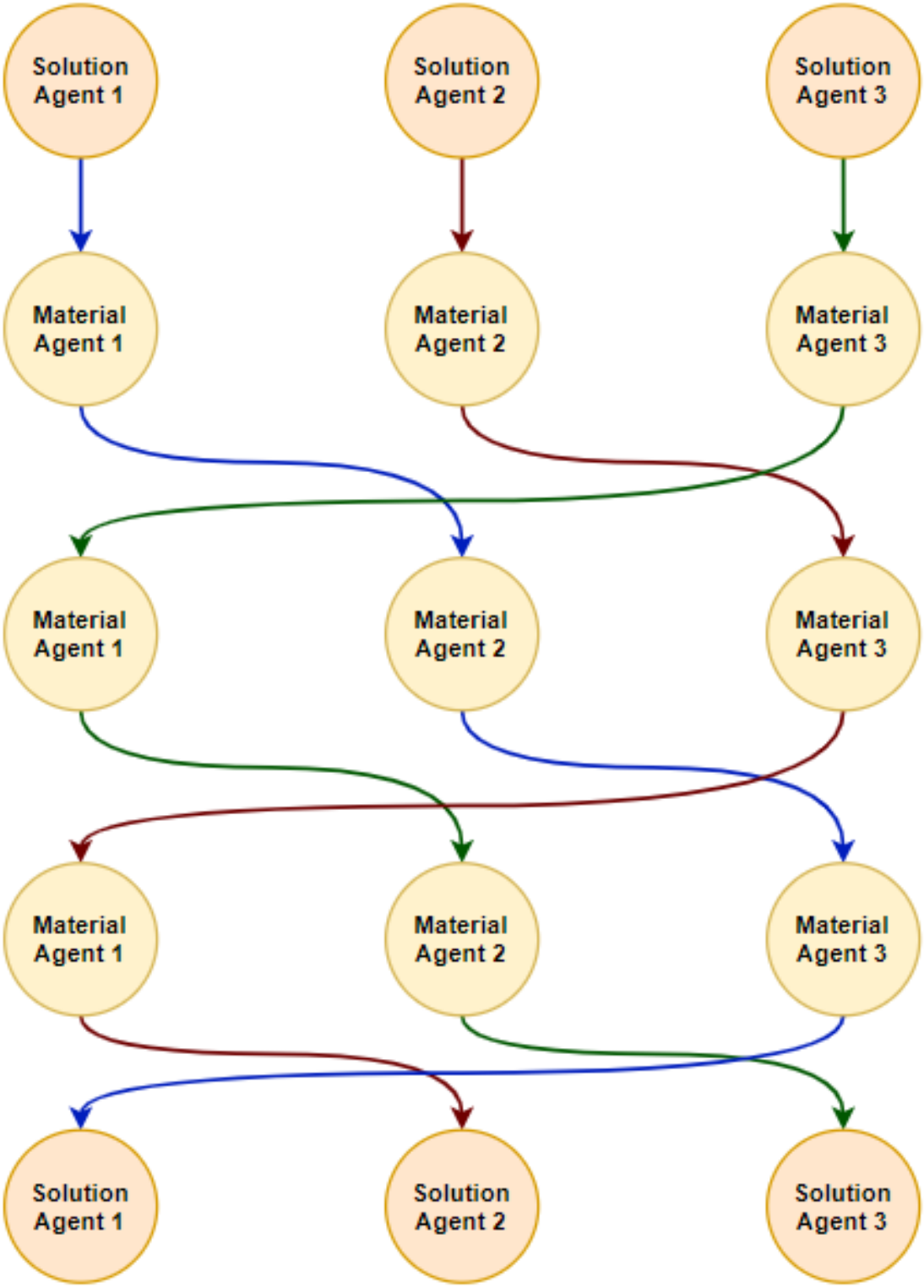

Figure 4.4: Three Solution Agents with three Material Agents

4.3.2 Material Agent

The Material Agent starts working when it receives a message either from a solution agent or another material agent. This agent's goal is to allocate its jobs in the provided scenario and relay it together with that scenario together with the ordering it received. The agent that will receive this message should be the next Material Agent in the ordering or to the Solution Agent that started the process if there is no next Agent Material. Note that it is in these agents that the actual scheduling process will occur.

The jobs in a Material Agent are scheduling from the first in the sequence to the last (forward scheduling), this method aims to reduce tardiness (while backward scheduling aims to reduce the earliness). This recursive call to allocate a job in the material receives the Scenario modified by the previous material (or the original scenario received by the Agent if it's the first job) and the job definition. To ensure that no job starts before the previous in the sequence finished, a variable in the material called Earliest Possible Date is maintained.

The first step is to determine if the current job requires a machine or not. To do this we check if the Job variable "Schedule Job Type" is set to one. If the job does not require a machine we simply have to update the Earliest Possible Date and move to the next job (or finish).

If our job does require a machine (this is the most common), we have to decide when and to which machine we will allocate it. We start by deciding what is the best machine to do the allocation. We do so by generating a proposal from each machine in the scenario. This results in a list of proposals, which specifies the machine and the starting time. Furthermore, this list is ordered by the starting date, using the last performed job date as a tiebreaker to ensure the jobs are well distributed between machines. Each job will only test the first proposal, except in cases the allocation to that proposal proves to be unfeasible.

After knowing the machine and the starting time proposed, we now need to perform the actual allocation. First, we check if the job correctly fits the machine (because in our environment jobs are already created with this in mind, this is more of a sanity check). We then try to check if we can insert our job together with any existing jobs in the machine.

If we cannot append our job to an existing allocation, we need to create a new one. The first thing we do is determine what will be the Processing Time and the Setup Time in this machine with this starting date. The Processing Time can be found by resolving the Processing Time table in the machine by providing the Product Group, Product and service. The Setup Time can be found by finding what will be the previous job if we start at the proposed starting time, and we then resolve the setup matrix (or use the default if there is no setup matrix entry for that pair). The sum of these values gives us the TotalProcessingTime (ProcessingTime + Setup).

After calculating the TotalProcessingTime, the final step is to determine what will be the planned end date for this job. Because there are three different ways a job can interact with the machine, we have three different strategies:

- The Resource can Process in Non Working Time. In this situation we simply add the Total Processing Time to the Start Date.

- The Job can Spread Across Working Times. If a job can spread across working times, we check if the working time where the starting time belongs is enough to allocate the job. If it is, we simply allocate it. If not, we occupy that working time completely and recursively try to allocate the remaining time to the next Working Time.

- None of the above. We try to allocate the job to the Working Time where the starting time belongs. If we fail the allocation fails and we move to the next proposal.

After determining the end time, we still need to determine if the allocation is viable by performing two checks:

- If the job we've placed isn't the last one in the machine, we may be affecting the setup time of a job that is scheduled for later. If this happens we need to adjust it to the new parameters.

- After doing this adjustment, and also because of the Process on Non Working time functionality, we have to check for collision with other jobs. If a collision happens, the allocation fails.

When we arrive to the last job, this function ends and the material is ready to transmit a message to the next agent. Note that if at any time a job fails to be allocated to any proposal, the material allocation fails. In this case the process continues with the jobs until the one that failed.

Chapter 5

Experiments and Results Analysis

In this Chapter we will describe the scenario that was used to evaluate our implementation, and present and discuss the results obtained.

5.1 Scenario Description

To test our solution, we've built a scenario that provided enough complexity while being fast enough to make the testing and evaluation possible. In table 5.1, we can see the details of this scenario, and then in each Subsection of the results, we can see more specific characteristics of each case.

Feature	**Details**
Materials	50
Machines	21
Jobs	350
Products	4
Steps	Each material requires 7 steps. Each step has 3 machines capable of performing it.
Setup Time	In step 3, one of the products has a 1 hour setup time moving to a different product and a 3 hour setup time moving from a different product.
Scheduling Start	2019-06-27 at 00:00
Due Date	2019-06-27 at 00:00 for every material
Processing Times	Processing time varies between steps but no between materials.
Working Time	Jobs can be spread across Working Times and resources can't process in Non Working Time.
Breaks	The scenario features one lunch break of two hours.

Table 5.1: Test Scenario

5.2 Results Analysis

In the following subsections, we will describe what are we testing, and show scenarios with different parameters to illustrate our results.

In every scenario we specify a coefficient for the total tardiness, total makespan (sum of the makespan of each job), and for the total setup time.

We also show the results using the modes fast, and dispatch, as well as the results in each of the five iterations of the genetic mode.

In each scenario we show the total tardiness, makespan and setup time, as well as the time it took for the first and last Solution Agents to complete.

5.2.1 Total Scheduling

In total scheduling we schedule all our jobs while considering that every machine is working properly.

The minimum total setup time in this case is 3 hours, which corresponds to each machine of step 3 of the materials, making a single transition in the best direction. A setup lower than 3 would require stacking all the products in one machine, but because we are doing forward scheduling that is not possible as it would create an imbalance in the job distribution.

Figure 5.1 illustrates one output from Total Scheduling.

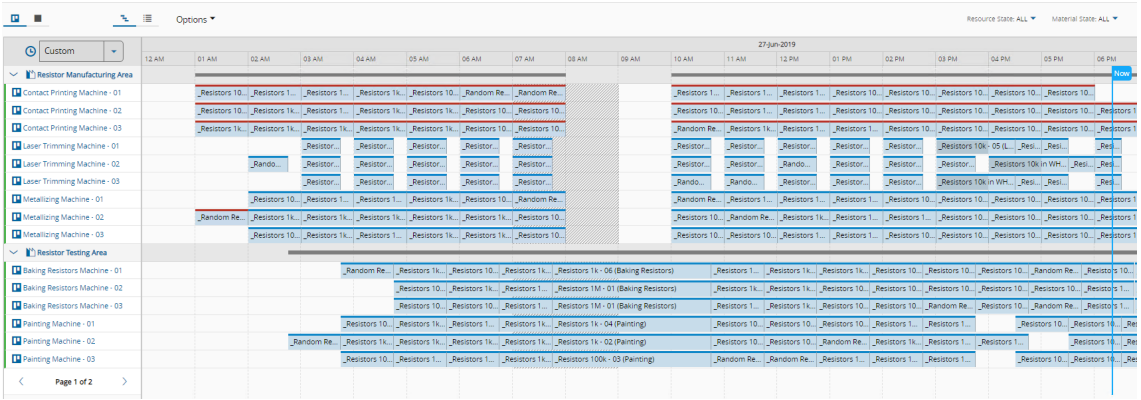

Figure 5.1: Full Schedule

5.2.1.1 Scenario 1

In the first scenario we set the setup coefficient to the maximum and the others to zero, so that we can evaluate the program's capacity to minimize the setup time with no interference.

In table 5.2 we can see the parameters.

Parameter	Value
Tardiness	0
Makespan	0
Setup Time	1

Table 5.2: Test Scenario 1 Parameters

In table 5.3 we can see the results.

Instance	Tardiness	Makespan	Setup Time	First Response	Last Response
Fast	2197 hours, 44 hours/material	600 hours, 12 hours/material	9 hours	<1 sec	<1 sec
Dispatch	2197 hours, 44 hours/material	600 hours, 12 hours/material	9 hours	5 secs	11 secs
Genetic 1st Generation	2166 hours, 43.3 hours/material	569 hours, 11.4 hours/material	3 hours	6 secs	7 secs
Genetic 2nd Generation	2166 hours, 43.3 hours/material	569 hours, 11.4 hours/material	3 hours	4 secs	7 secs
Genetic 3rd Generation	2166 hours, 43.3 hours/material	569 hours, 11.4 hours/material	3 hours	4 secs	7 secs
Genetic 4th Generation	2166 hours, 43.3 hours/material	569 hours, 11.4 hours/material	3 hours	4 secs	7 secs
Genetic 5th Generation	2166 hours, 43.3 hours/material	569 hours, 11.4 hours/material	3 hours	4 secs	7 secs

Table 5.3: Test Scenario 1 Results

We can see that the best solution for both the target KPI and the other KPIs was attained using the Genetic Mode, with the Dispatch Mode and Fast Mode solutions getting the same results.

The Fast mode was the fastest, followed by the Dispatch Mode and then the Genetic Mode. Noting that the execution time of each generation of the Genetic algorithm was lower than the Dispatch mode.

Because the result from the first generation was already optimal there were no improvements over time for the Genetic Mode.

5.2.1.2 Scenario 2

In the second scenario we set the makespan coefficient to the maximum and the others to zero, so that we can evaluate the program's capacity to minimize the total makespan of materials with no interference.

In table 5.4 we can see the parameters.

Parameter	Value
Tardiness	0
Makespan	1
Setup Time	0

Table 5.4: Test Scenario 2 Parameters

In table 5.5 we can see the results.

Instance	Tardiness	Makespan	Setup Time	First Response Time	Last Response Time
Fast	2197 hours, 44 hours/material	600 hours, 12 hours/material	9 hours	<1 sec	<1 sec
Dispatch	2165 hours, 43.3 hours/material	567 hours / 11.35 hours/material	10 hours	9 secs	14 secs
Genetic 1st Generation	2152 hours, 43 hours/material	555 hours / 11.1 hours/material	3 hours	5 secs	8 secs
Genetic 2nd Generation	2140 hours / 42.8 hours/material	543 hours / 10.9 hours/material	14 hours	6 secs	8 secs
Genetic 3rd Generation	2127 hours / 42.5 hours/material	530 hours / 10.6 hours/material	21 hours	4 secs	7 secs
Genetic 4th Generation	2127 hours / 42.5 hours/material	530 hours / 10.6 hours/material	21 hours	5 secs	7 secs
Genetic 5th Generation	2127 hours / 42.5 hours/material	530 hours / 10.6 hours/material	21 hours	5 secs	8 secs

Table 5.5: Test Scenario 2 Results

We can see that the best solution for both the target KPI and the other KPIs was attained using the Genetic Mode, with the Dispatch Mode getting better results than the Fast Mode.

The Fast mode was the fastest, followed by the Dispatch Mode and then the Genetic Mode.Noting that the execution time of each generation of the Genetic algorithm was lower than the Dispatch mode.

The Genetic Mode saw improvements over the first three generation, but no improvements in the last two.

5.2.1.3 Scenario 3

In the third scenario we set the tardiness coefficient to the maximum and the others to zero, so that we can evaluate the program's capacity to minimize the total tardiness of materials with no interference.

In table 5.6 we can see the parameters.

Parameter	**Value**
Tardiness	1
Makespan	0
Setup Time	0

Table 5.6: Test Scenario 3 Parameters

In table 5.7 we can see the results.

Instance	**Tardiness**	**Makespan**	**Setup Time**	**First Response Time**	**Last Response Time**
Fast	2197 hours, 44 hours/material	600 hours, 12 hours/material	9 hours	<1 sec	<1 sec
Dispatch	2165 hours, 43.3 hours/material	567 hours / 11.35 hours/material	10 hours	9 secs	14 secs
Genetic 1st Generation	2148 hours, 43 hours/material	551 hours, 11 hours/material	3 hours	5 secs	7 secs
Genetic 2nd Generation	2148 hours, 43 hours/material	551 hours, 11 hours/material	3 hours	4 secs	6 secs
Genetic 3rd Generation	2114 hours, 42.3 hours/material	517 hours, 10.3 hours/material	15 hours	4 secs	6 secs
Genetic 4th Generation	2114 hours, 42.3 hours/material	517 hours, 10.3 hours/material	15 hours	4 secs	6
Genetic 5th Generation	2112 hours, 42.2 hours/material	515 hours, 10.3 hours/material	14 hours	4 secs	6 secs

Table 5.7: Test Scenario 3 Results

We can see that the best solution for both the target KPI and the other KPIs was attained using the Genetic Mode, with the Dispatch Mode getting better results than the Fast Mode.

The Fast mode was the fastest, followed by the Dispatch Mode and then the Genetic Mode. Noting that the execution time of each generation of the Genetic algorithm was lower than the Dispatch mode.

The Genetic Mode saw improvements from the second to the third generation and from the fourth to the fifth.

It is also of note, that this particular execution found better makespan solutions than the one in the last scenario. This happens because the KPIs are not independent from each other and due to the stochastic nature of the genetic algorithm.

5.2.1.4 Scenario 4

In the fourth scenario we gave equal weight to every objective, so that we can evaluate the program's capacity to schedule under a multi-objective goal.

In table 5.8 we can see the parameters.

Parameter	**Value**
Tardiness	0.34
Makespan	0.33
Setup Time	0.33

Table 5.8: Test Scenario 4 Parameters

In table 5.9 we can see the results.

Instance	**Tardiness**	**Makespan**	**Setup Time**	**First Response Time**	**Last Response Time**
Fast	2197 hours, 44 hours/material	600 hours, 12 hours/material	9 hours	<1 sec	<1 sec
Dispatch	2197 hours, 44 hours/material	600 hours, 12 hours/material	9 hours	9 secs	15 secs
Genetic 1st Generation	2150 hours, 43 hours/material	553 hours, 11 hours/material	3 hours	6 secs	7 secs
Genetic 2nd Generation	2150 hours, 43 hours/material	553 hours, 11 hours/material	3 hours	6 secs	7 secs
Genetic 3rd Generation	2146 hours, 42.9 hours/material	549 hours, 11 hours/material	3 hours	5 secs	6 secs
Genetic 4th Generation	2146 hours, 42.9 hours/material	549 hours, 11 hours/material	3 hours	5 secs	7 secs
Genetic 5th Generation	2146 hours, 42.9 hours/material	549 hours, 11 hours/material	3 hours	5 secs	6 secs

Table 5.9: Test Scenario 4 Results

We can see that the best solution for both the target KPI and the other KPIs was attained using the Genetic Mode, with the Dispatch Mode and Fast Mode solutions getting the same results.

The Fast mode was the fastest, followed by the Dispatch Mode and then the Genetic Mode. Noting that the execution time of each generation of the Genetic algorithm was lower than the Dispatch mode.

The Genetic Mode saw improvements from the second to the third generation.

It is also of note, that due the setup time ended up being the dominant objective due to having larger percentage variations.

5.2.2 Machine Failure and Total Scheduling

In total scheduling under machine failure, we schedule all our jobs but we do not consider the failed machine. We can see that our KPIs are much worse in this case.

Figure 5.2 illustrates one output from Total Scheduling. We can see that Laser Trimming Machine - 01 is gone and that the other two have more load.

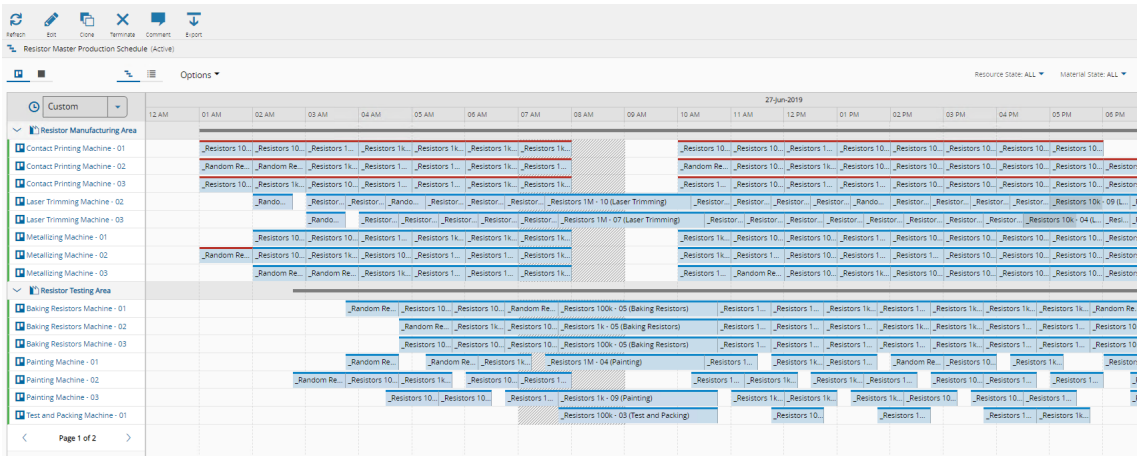

Figure 5.2: Full Schedule under Machine Failure

5.2.2.1 Scenario 5

In the fifth scenario we set the setup coefficient to the maximum and the others to zero, so that we can evaluate the program's capacity to minimize the setup time with no interference.

In table 5.10 we can see the parameters.

Parameter	Value
Tardiness	0
Makespan	0
Setup Time	1

Table 5.10: Test Scenario 5 Parameters

In table 5.11 we can see the results.

Instance	**Tardiness**	**Makespan**	**Setup Time**	**First Response Time**	**Last Response Time**
Fast	2206 hours, 44.1 hours/material	609 hours, 12.2 hours/material	6 hours	<1 sec	<1 sec
Dispatch	2206 hours, 44.1 hours/material	609 hours, 12.2 hours/material	6 hours	8 secs	15 secs
Genetic 1st Generation	2159 hours, 43.2 hours/material	562 hours, 11.2 hours/material	2 hours	6 secs	7 secs
Genetic 2nd Generation	2159 hours, 43.2 hours/material	562 hours, 11.2 hours/material	2 hours	5 secs	7 secs
Genetic 3rd Generation	2159 hours, 43.2 hours/material	562 hours, 11.2 hours/material	2 hours	4 secs	6 secs
Genetic 4th Generation	2159 hours, 43.2 hours/material	562 hours, 11.2 hours/material	2 hours	4 secs	6 secs
Genetic 5th Generation	2159 hours, 43.2 hours/material	562 hours, 11.2 hours/material	2 hours	4 secs	7 secs

Table 5.11: Test Scenario 5 Results

We can see that the best solution for both the target KPI and the other KPIs was attained using the Genetic Mode, with the Dispatch Mode and Fast Mode solutions getting the same results.

The Fast mode was the fastest, followed by the Dispatch Mode and then the Genetic Mode. Noting that the execution time of each generation of the Genetic algorithm was lower than the Dispatch mode.

Because the result from the first generation was already optimal there were no improvements over time for the Genetic Mode.

5.2.2.2 Scenario 6

In the sixth scenario we set the makespan coefficient to the maximum and the others to zero, so that we can evaluate the program's capacity to minimize the total makespan of materials with no interference.

In table 5.12 we can see the parameters.

Parameter	Value
Tardiness	0
Makespan	1
Setup Time	0

Table 5.12: Test Scenario 6 Parameters

In table 5.13 we can see the results.

Instance	Tardiness	Makespan	Setup Time	First Response Time	Last Response Time
Fast	2206 hours, 44.1 hours/material	609 hours, 12.2 hours/material	6 hours	<1 sec	<1 sec
Dispatch	2182 hours, 43.7 hours/material	585 hours, 11.7 hours/material	7 hours	6 secs	10 secs
Genetic 1st Generation	2151 hours, 43 hours/material	554 hours, 11 hours/material	2 hours	5 secs	6 secs
Genetic 2nd Generation	2127 hours, 42.5 hours/material	530 hours, 10.6 hours/material	10 hours	4 secs	6 secs
Genetic 3rd Generation	2127 hours, 42.5 hours/material	530 hours, 10.6 hours/material	10 hours	4 secs	6 secs
Genetic 4th Generation	2127 hours, 42.5 hours/material	530 hours, 10.6 hours/material	10 hours	4 secs	6 secs
Genetic 5th Generation	2127 hours, 42.5 hours/material	530 hours, 10.6 hours/material	10 hours	5 secs	6 secs

Table 5.13: Test Scenario 6 Results

We can see that the best solution for both the target KPI and the other KPIs was attained using the Genetic Mode, with the Dispatch Mode getting better results than the Fast Mode.

The Fast mode was the fastest, followed by the Dispatch Mode and then the Genetic Mode.Noting that the execution time of each generation of the Genetic algorithm was lower than the Dispatch mode.

The Genetic Mode saw improvements from the second to the third generation.

5.2.2.3 Scenario 7

In the seventh scenario we set the tardiness coefficient to the maximum and the others to zero, so that we can evaluate the program's capacity to minimize the total tardiness of materials with no interference.

In table 5.14 we can see the parameters.

Parameter	Value
Tardiness	1
Makespan	0
Setup Time	0

Table 5.14: Test Scenario 7 Parameters

In table 5.15 we can see the results.

Instance	Tardiness	Makespan	Setup Time	First Response Time	Last Response Time
Fast	2206 hours, 44.1 hours/material	609 hours, 12.2 hours/material	6 hours	<1 sec	<1 sec
Dispatch	2182 hours, 43.7 hours/material	585 hours, 11.7 hours/material	7 hours	6 secs	10 secs
Genetic 1st Generation	2153 hours, 43 hours/material	556 hours, 11.1 hours/material	2 hours	4 secs	6 secs
Genetic 2nd Generation	2146 hours, 43 hours/material	549 hours, 11 hours/material	10 hours	4 secs	6 secs
Genetic 3rd Generation	2118 hours, 42.4 hours/material	521 hours, 10.4 hours/material	8 hours	4 secs	6 secs
Genetic 4th Generation	2118 hours, 42.4 hours/material	521 hours, 10.4 hours/material	8 hours	5 secs	6 secs
Genetic 5th Generation	2118 hours, 42.4 hours/material	521 hours, 10.4 hours/material	8 hours	5 secs	6 secs

Table 5.15: Test Scenario 7 Results

We can see that the best solution for both the target KPI and the other KPIs was attained using the Genetic Mode, with the Dispatch Mode getting better results than the Fast Mode.

The Fast mode was the fastest, followed by the Dispatch Mode and then the Genetic Mode. Noting that the execution time of each generation of the Genetic algorithm was lower than the Dispatch mode.

The Genetic Mode saw improvements over the first three generation, but no improvements in the last two.

It is also of note, that this particular execution found better makespan solutions than the one in the last scenario. This happens because the KPIs are not independent from each other and due to the stochastic nature of the genetic algorithm.

5.2.2.4 Scenario 8

In the eight scenario we gave equal weight to every objective, so that we can evaluate the program's capacity to schedule under a multi-objective goal.

In table 5.16 we can see the parameters.

Parameter	Value
Tardiness	0.34
Makespan	0.33
Setup Time	0.33

Table 5.16: Test Scenario 8 Parameters

In table 5.17 we can see the results.

Instance	Tardiness	Makespan	Setup Time	First Response Time	Last Response Time
Fast	2206 hours, 44.1 hours/material	609 hours, 12.2 hours/material	6 hours	<1 sec	<1 sec
Dispatch	2206 hours, 44.1 hours/material	609 hours, 12.2 hours/material	6 hours	6 secs	11 secs
Genetic 1st Generation	2149 hours, 43 hours/material	552 hours, 11 hours/material	2 hours	5 secs	6 secs
Genetic 2nd Generation	2149 hours, 43 hours/material	552 hours, 11 hours/material	2 hours	4 secs	6 secs
Genetic 3rd Generation	2149 hours, 43 hours/material	552 hours, 11 hours/material	2 hours	4 secs	6 secs
Genetic 4th Generation	2145 hours, 42.9 hours/material	548 hours, 10.96 hours/material	2 hours	5 secs	6 secs
Genetic 5th Generation	2145 hours, 42.9 hours/material	548 hours, 10.96 hours/material	2 hours	5 secs	6 secs

Table 5.17: Test Scenario 8 Results

We can see that the best solution for both the target KPI and the other KPIs was attained using the Genetic Mode, with the Dispatch Mode and Fast Mode solutions getting the same results.

The Fast mode was the fastest, followed by the Dispatch Mode and then the Genetic Mode. Noting that the execution time of each generation of the Genetic algorithm was lower than the Dispatch mode.

The Genetic Mode saw improvements from the third to the fourth generation.

It is also of note, that due the setup time ended up being the dominant objective due to having larger percentage variations.

5.2.3 Machine Failure and Partial Scheduling

In machines failure and partial scheduling we are only scheduling materials affected by the machine failure, without moving the others. To ensure consistency our base scenario was always scenario 4. We reschedule 16 materials, which as expected correspond to about one third of the total, and the KPI values are only for these 16 materials.

5.2.3.1 Scenario 9

In the ninth scenario we gave equal weight to every objective, so that we can evaluate the program's capacity to schedule under a multi-objective goal.

In table 5.18 we can see the parameters.

Parameter	Value
Tardiness	0.34
Makespan	0.33
Setup Time	0.33

Table 5.18: Test Scenario 9 Parameters

In table 5.19 we can see the results.

We can see that the best solution for both the target KPI and the other KPIs was attained using the Genetic Mode, with the Dispatch Mode getting better results than the Fast Mode.

The Fast mode was the fastest, followed by the Dispatch Mode and then the Genetic Mode. Noting that the execution time of each generation of the Genetic algorithm was lower than the Dispatch mode.

The Genetic Mode saw improvements from the third to the fourth generation.

It is also of note, that due the setup time ended up being the dominant objective due to having larger percentage variations.

5.3 Conclusions

Our results show that, as expected, the fast scheduling mode performs the fastest while providing inferior results. Based on that, we conclude that the fast mode should only be used in a situation where getting a decent schedule fast is crucial. The dispatch mode proved not to be the best

Instance	**Tardiness**	**Makespan**	**Setup Time**	**First Response Time**	**Last Response Time**
Fast	818 hours, 51.2 hours/material	313 hours, 19.6 hours/material	0 hours	<1 sec	<1 sec
Dispatch	817 hours, 51.1 hours/material	312 hours, 19.5 hours/material	0 hours	<1 sec	<1 sec
Genetic 1st Generation	811 hours, 51 hours/material	306 hours, 19 hours/material	0 hours	<1 sec	<1 sec
Genetic 2nd Generation	811 hours, 51 hours/material	306 hours, 19 hours/material	0 hours	<1 sec	<1 sec
Genetic 3rd Generation	811 hours, 51 hours/material	306 hours, 19 hours/material	0 hours	<1 sec	<1 sec
Genetic 4th Generation	808 hours, 50.5 hours/material	303 hours, 19 hours/material	0 hours	<1 sec	<1 sec
Genetic 5th Generation	808 hours, 50.5 hours/material	303 hours, 19 hours/material	0 hours	<1 sec	<1 sec

Table 5.19: Test Scenario 9 Results

solution for nearly any case because it was both slower and provided worse results than the first generation of the genetic algorithm in every scenario. Because the dispatch mode size depends on the number of materials, we conclude that the only use for dispatch mode must be for some low load scenarios. And lastly, we were able to conclude that the best solutions among all three modes were found using the genetic algorithm. One thing that is important to note is that we've run every agent on a single computer, have we ran each of the 30 possible material orderings in parallel the time could be comparable to the fast mode. It is also of note, that besides having a satisfactory first solution, the genetic mode also saw clear improvements between generations, in most scenarios. About the scenarios, we can see that the scheduler was able to provide a solution for total and partial rescheduling, with or without specifying a failed machine. One thing that was the results also made clear, is that when an objective is subject to higher percentage variations (in this case the setup time), if we give the same factors to every goal, that goal is going to have a heavier weight.

Chapter 6

Conclusions and Future Work

In this chapter, we will present the final remarks and briefly discuss further work.

6.1 Main Contributions

The study of the JSP has caught for many years the attention of various researches, and although there were clear advancements and innovative ideas, it is noticeable that there is a lack of a definitive solution that covers every situation. With this dissertation, we have:

- Helped to bridge the gap between the scientific and industrial community by demonstrating how we've applied the concepts from our Literature Review to our solution implementation.
- Provided an inspiring agent-based implementation of the genetic algorithm for JSP that others can extend or use to complete their solutions.
- Helped describe the problem as a multi-objective, multi-dimension environment.
- Provided scientific content about the complex domain of rescheduling.
- This dissertation also originated a paper which is a summary of the contents found in this dissertation.

6.2 Future Work

Although our implementation tried to be the as whole as possible, we believe there is still room for improvement. Some of the future work we would like to see in this field is:

- The integration of resources besides machines, namely employees and tools, in our model, or a similar one.
- Data about how having different scheduling strategies in each Agent Material (or equivalent) would affect the results.

- How custommer preferences can affect the scheduling strategy at the Agent Material level (as opposed to having a universal strategy for every resource).
- How other popular heuristics used in artificial intelligence field, such as Ant Colony Optimization would compare to the Genetic Algorithm.
- What would be the viability of improving individuals at the end of each generation with a local search strategy such as Beam Search or Tabu Search. And how would that affect the results.

References

[ABZ88] Joseph Adams, Egon Balas, and Daniel Zawack. *The Shifting Bottleneck Procedure for Job Shop Scheduling*, volume 34. 1988.

[Ake56] Sheldon B. Akers. Letter to the editor—a graphical approach to production scheduling problems. *Operations Research*, 4(2):244–245, 1956.

[Apt03] Krzysztof Apt. *Principles of Constraint Programming*. Cambridge University Press, Cambridge, 2003.

[BAMA18] D. C. Bissoli, W. A. S. Altoe, G. R. Mauri, and A. R. S. Amaral. A simulated annealing metaheuristic for the bi-objective flexible job shop scheduling problem. In *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*, pages 1–6, 2018.

[BB19] Julia C. Bendul and Henning Blunck. The design space of production planning and control for industry 4.0. *Computers in Industry*, 105:260–272, 2019.

[BBB19] A. Bekkar, G. Belalem, and B. Beldjilali. Iterated greedy insertion approaches for the flexible job shop scheduling problem with transportation times constraint. *International Journal of Manufacturing Research*, 14(1):43–66, 2019.

[BCG07] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. John Wiley & Sons, Inc., 2007.

[BFR15] E. G. Birgin, J. E. Ferreira, and D. P. Ronconi. List scheduling and beam search methods for the flexible job shop scheduling problem with sequencing flexibility. *European Journal of Operational Research*, 247(2):421–440, 2015.

[BJS94] Peter Brucker, Bernd Jurisch, and Bernd Sievers. A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics*, 49(1):107–127, 1994.

[Blu05] Christian Blum. *Beam-ACO - Hybridizing ant colony optimization with beam search: An application to open shop scheduling*, volume 32. 2005.

[BLV95] Egon Balas, Jan Karel Lenstra, and Alkis Vazacopoulos. The one-machine problem with delayed precedence constraints and its use in job shop scheduling. *Management Science*, 41(1):94–109, 1995.

[BPR] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade - a fipa-compliant agent framework.

[Bur84] D. N. Burghes. "sequencing and scheduling." by s. french. (ellis horwood ltd., 1982.) [pp. 245.]. *International Journal of Production Research*, 22(3):532–532, 1984.

[BV98] Egon Balas and Alkis Vazacopoulos. Guided local search with shifting bottleneck for job shop scheduling. *Management Science*, 44(2):262–275, 1998.

[BW65] G. H. Brooks and C. R. White. *An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem*, volume 16. 1965.

[CAG+18] R. Chaudhary, G. S. Aujla, S. Garg, N. Kumar, and J. J. P. C. Rodrigues. Sdn-enabled multi-attribute-based secure communication for smart grid in iiot environment. *IEEE Transactions on Industrial Informatics*, 14(6):2629–2640, 2018.

[CBCJO19] I. Castelo-Branco, F. Cruz-Jesus, and T. Oliveira. Assessing industry 4.0 readiness in manufacturing: Evidence for the european union. *Computers in Industry*, 107:22–32, 2019.

[CDM91] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. *Distributed Optimization by Ant Colonies*. 1991.

[CGT96] Runwei Cheng, Mitsuo Gen, and Yasuhiro Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithms—i. representation. *Computers & industrial engineering*, 30(4):983–997, 1996.

[CGT99] Runwei Cheng, Mitsuo Gen, and Yasuhiro Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithms, part ii: hybrid genetic search strategies. *Computers & Industrial Engineering*, 36(2):343–364, 1999.

[Cre95] Daniel Crevier. Daniel crevier. ai: The tumultuous history of the search for artificial intelligence. ny: Basic books, 1993. 432 pp. (reviewed by charles fair). *Journal of the History of the Behavioral Sciences*, 31(3):273–278, 1995.

[CUB13] Banu Calis Uslu and Serol Bulkan. *A research survey: review of AI solution strategies of job shop scheduling problem*, volume 26. 2013.

[Das14] Partha Das. Ant colony optimization presentation, 2014.

[DAS19] S. S. Dewi, A. Andriansyah, and S. Syahriza. Optimization of flow shop scheduling problem using tabu search algorithm: A case study. In *IOP Conference Series: Materials Science and Engineering*, volume 506, 2019.

[Dav85] Lawrence Davis. Job shop scheduling with genetic algorithms. In *Proceedings of an international conference on genetic algorithms and their applications*, volume 140, 1985.

[DFP95] Didier Dubois, Hélène Fargier, and Henri Prade. Fuzzy constraints in job-shop scheduling. *Journal of Intelligent Manufacturing*, 6(4):215–234, 1995.

[DH14] R. Drath and A. Horch. Industrie 4.0: Hit or hype? [industry forum]. *IEEE Industrial Electronics Magazine*, 8(2):56–58, 2014.

[dM19] R. de Matta. Minimizing the total waiting time of intermediate products in a manufacturing process. *International Transactions in Operational Research*, 26(3):1096–1117, 2019.

[DMC91] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: An autocatalytic optimizing process. 1991.

[DPL93] S. Dauzere-Peres and J. B. Lasserre. A modified shifting bottleneck procedure for job-shop scheduling. *International Journal of Production Research*, 31(4):923–932, 1993.

[DT93] Mauro Dell'Amico and Marco Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41(3):231–252, 1993.

[FB91] E. Falkenauer and S. Bouffouix. A genetic algorithm for job shop. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 824–829 vol.1, 1991.

[FF95] Carlos M Fonseca and Peter J Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, 3(1):1–16, 1995.

[FIP05] FIPA. Fipa specifications, 2005.

[FM83] Fox and Mark. *Constraint-directed search : a case study of job-shop scheduling / Mark S. Fox*. 1983.

[FYPT88a] Simon Foo Yoon-Pin and Takefuji. Integer linear programming neural networks for job-shop scheduling. In *IEEE 1988 International Conference on Neural Networks*, pages 341–348 vol.2, 1988.

[FYPT88b] Simon Foo Yoon-Pin and Takefuji. Stochastic neural networks for solving job-shop scheduling. i. problem representation. In *IEEE 1988 International Conference on Neural Networks*, pages 275–282 vol.2, 1988.

[GLLK79] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. *Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey*, volume 5, pages 287–326. Elsevier, 1979.

[GM16] T. E. E. Goncalo and D. C. Morais. Agent-based negotiation protocol for selecting transportation providers in a retail company. In *Proceedings - 2015 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2015*, pages 263–267, 2016.

[GS78] Teofilo Gonzalez and Sartaj Sahni. *Flowshop and Jobshop Schedules: Complexity and Approximation*, volume 26. 1978.

[GSW18] O. Gholami, Y. N. Sotskov, and F. Werner. A genetic algorithm for hybrid job-shop scheduling problems with minimizing the makespan or mean flow time. *Journal of Advanced Manufacturing Systems*, 17(4):461–486, 2018.

[HA82] N. Hefetz and I. Adiri. An efficient optimal algorithm for the two-machines unit-time jobshop schedule-length problem. *Math. Oper. Res.*, 7(3):354–360, 1982.

[HC98] Yi-Feng Hung and Ing-Ren Chen. A simulation study of dispatch rules for reducing flow times in semiconductor wafer fabrication. *Production Planning & Control*, 9(7):714–722, 1998.

[HJT94] Johann Hurink, Bernd Jurisch, and Monika Thole. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, 15(4):205–215, 1994.

[HL08] Kuo-Ling Huang and Ching-Jong Liao. *Ant colony optimization combined with taboo search for the job shop scheduling problem*, volume 35. 2008.

[Hol92] John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[HY17] Rong-Hwa Huang and Tung-Han Yu. An effective ant colony optimization algorithm for multi-objective job-shop scheduling with equal-size lot-splitting. *Applied Soft Computing*, 57:642–656, 2017.

[HY19] X. Huang and L. Yang. A hybrid genetic algorithm for multi-objective flexible job shop scheduling problem considering transportation time. *International Journal of Intelligent Computing and Cybernetics*, 12(2):154–174, 2019.

[IBH+16] D Imkamp, J Berthold, Michael Heizmann, K Kniel, Eberhard Manske, M Peterek, Robert Schmitt, J Seidler, and Klaus-Dieter Sommer. Challenges and trends in manufacturing measurement technology - the "industrie 4.0" concept. *Journal of Sensors and Sensor Systems*, 5:325–335, 10 2016.

[IR11] Helga Ingimundardottir and Thomas Philip Runarsson. Supervised learning linear priority dispatch rules for job-shop scheduling. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 263–277. Springer Berlin Heidelberg, 2011.

[IS65] Edward Ignall and Linus Schrage. Application of the branch and bound technique to some flow-shop scheduling problems. *Operations Research*, 13(3):400–412, 1965.

[Jac56] James R. Jackson. An extension of johnson's results on job idt scheduling. *Naval Research Logistics Quarterly*, 3(3):201–203, 1956.

[JAD19a] JADE. Java agent development framework, 2019.

[JAD19b] JADE. Java agent development framework api, 2019.

[JAI17] SHUBHAM JAIN. Introduction to genetic algorithm & their application in data science, 2017.

[JM98] A. S. Jain and S. Meeran. Job-shop scheduling using neural networks. *International Journal of Production Research*, 36(5):1249–1272, 1998.

[JNF15] Amir Jalilvand-Nejad and Parviz Fattahi. A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem. *J. Intell. Manuf.*, 26(6):1085–1098, 2015.

[Joh54] S. M. Johnson. *Optimal Two and Three Stage Production Schedules With Set-Up Time Included*, volume 1. 1954.

[KGG18] Sachin S. Kamble, Angappa Gunasekaran, and Shradha A. Gawankar. Sustainable industry 4.0 framework: A systematic literature review identifying the current trends and future perspectives. *Process Safety and Environmental Protection*, 117:408–425, 2018.

[KGV83] Scott Kirkpatrick, Charles Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220 4598:671–80, 1983.

[Kin19] Ryan King. 7 benefits of ai in manufacturing, 2019.

[KLHGRKB77] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Bruckner. *Complexity of Machine Scheduling Problems*, volume 1. 1977.

[KMMC10] Artur Kuczapski, Mihai Micea, Laurentiu Maniu, and Vladimir Cretu. *Efficient Generation of Near Optimal Initial Populations to Enhance Genetic Algorithms for Job-Shop Scheduling*, volume 39. 2010.

[KYK18] Chen Kan, Hui Yang, and Soundar Kumara. Parallel computing and network analytics for fast industrial internet-of-things (iiot) machine information processing and condition monitoring. *Journal of Manufacturing Systems*, 46:282–293, 2018.

[LAL92] Peter J. M. van Laarhoven, Emile H. L. Aarts, and Jan Karel Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):113–125, 1992.

[LHW17] Guoping Li, Yun Hou, and Aizhi Wu. Fourth industrial revolution: technological drivers, impacts and coping methods. *Chinese Geographical Science*, 27(4):626–637, 2017.

[LM14] Lian Lian and Khaled Mesghouni. *Comparative study of heuristics algorithms in solving flexible job shop scheduling problem with condition based maintenance*, volume 7. 2014.

[LM18] Sunil Luthra and Sachin Kumar Mangla. Evaluating challenges to industry 4.0 initiatives for supply chain sustainability in emerging economies. *Process Safety and Environmental Protection*, 117:168–179, 2018.

[LOCS19] T. Lins, R. A. R. Oliveira, L. H. A. Correia, and J. S. Silva. Industry 4.0 retrofitting. In *Brazilian Symposium on Computing System Engineering, SBESC*, volume 2018-November, pages 8–15, 2019.

[Lom65] Z. A. Lomnicki. A "branch-and-bound" algorithm for the exact solution of the three-machine scheduling problem. *Journal of the Operational Research Society*, 16(1):89–100, 1965.

[Lu19] Y. Lu. Artificial intelligence: a survey on evolution, models, applications and future trends. *Journal of Management Analytics*, 6(1):1–29, 2019.

[Man] Critical Manufacturing. What is mes?

[Man60] Alan S. Manne. On the job-shop scheduling problem. *Operations Research*, 8(2):219–223, 1960.

[MB67] G. B. McMahon and P. G. Burton. Flow-shop scheduling with the branch-and-bound method. *Operations Research*, 15(3):473–481, 1967.

[mde16] mdestrian. Why iiot is different from iot, 2016.

[MF75] Graham McMahon and Michael Florian. *On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness*, volume 23. 1975.

[MKB+16] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2):621–641, 2016.

[MN17] Gonzalo Mejia and Karen Niño. A new hybrid filtered beam search algorithm for deadlock-free scheduling of flexible manufacturing systems using petri nets. *Computers & Industrial Engineering*, 108:165–176, 2017.

[MV18] Dimitris Mourtzis and Ekaterini Vlachou. A cloud-based cyber-physical system for adaptive shop-floor scheduling and condition-based maintenance. *Journal of Manufacturing Systems*, 47:179–198, 2018.

[MW59] Harvey M. Wagner. *An Integer Linear-Programming Model for Machine Scheduling*, volume 6. 1959.

[MY03] Hyeung-Sik Min and Yuehwern Yih. Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system. *International Journal of Production Research*, 41(16):3921–3941, 2003.

[MZ16] Pieter J. Mosterman and Justyna Zander. Industry 4.0 as a cyber-physical system study. *Software & Systems Modeling*, 15(1):17–29, 2016.

[MZMK11] H. S. Mirsanei, M. Zandieh, M. J. Moayed, and M. R. Khabbazi. A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, 22(6):965–978, 2011.

[OM88] Peng Si Ow and Thomas E. Morton. Filtered beam search in scheduling. *International Journal of Production Research*, 26(1):35–62, 1988.

[OP89] I. H. Osman and C. N. Potts. Simulated annealing for permutation flow-shop scheduling. *Omega*, 17(6):551–557, 1989.

[PAG18] Michela Piccarozzi, Barbara Aquilani, and Corrado Gatti. Industry 4.0 in management studies: A systematic literature review. *Sustainability*, 10(10):3821, 2018.

[PM00] Ferdinando Pezzella and Emanuela Merelli. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, 120(2):297–310, 2000.

[PME19] N. Pillay, B. T. Maharaj, and G. V. Eeden. Ai in engineering and computer science education in preparation for the 4th industrial revolution: A south african perspective. In *2018 World Engineering Education Forum - Global Engineering Deans Council, WEEF-GEDC 2018*, 2019.

[PRTS16] Midhun Paul, Radha Ramanan T, and R. Sridharan. *A multi-objective decision-making framework using preference selection index for assembly job shop scheduling problem*, volume 9. 2016.

[PT96] Erwin Pesch and Ulrich A. W. Tetzlaff. Constraint propagation based scheduling of job shops. *INFORMS Journal on Computing*, 8(2):144–157, 1996.

[PVW85] Chris Potts and Luk Van Wassenhove. *A Branch and Bound Algorithm for the Total Weighted Tardiness Problem*, volume 33. 1985.

[PZL12] Kyung-Joon Park, Rong Zheng, and Xue Liu. Cyber-physical systems: Milestones and research challenges. *Computer Communications*, 36(1):1–7, 2012.

[QMQ+19] Y. J. Qu, X. G. Ming, S. Q. Qiu, Z. W. Liu, X. Y. Zhang, and Z. T. Hou. A framework for smart manufacturing systems based on the stakeholders' value. In *Proceedings of the 2018 IEEE International Conference on Advanced Manufacturing, ICAM 2018*, pages 239–242, 2019.

[RBW06] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., 2006.

[RH99] Chandrasekharan Rajendran and Oliver Holthaus. *Comparative study of dispatching rules in dynamic flowshops and jobshops*, volume 116. 1999.

[RK19] Rajan and V. Kumar. Integration of dispatch rules for jssp: A learning approach. In *Advances in Intelligent Systems and Computing*, volume 742, pages 619–627. 2019.

[ROL18] Nelson Rodrigues, Eugénio Oliveira, and Paulo Leitão. Decentralized and on-the-fly agent-based service reconfiguration in manufacturing systems. *Computers in Industry*, 101:81–90, 2018.

[Ros01] O. Rose. The shortest processing time first (sptf) dispatch rule and some variants in semiconductor manufacturing. In *Proceeding of the 2001 Winter Simulation Conference (Cat. No.01CH37304)*, volume 2, pages 1220–1224 vol.2, 2001.

[RPKKMH19] N. Rafiee Parsa, T. Keshavarz, B. Karimi, and S. M. Moattar Husseini. A hybrid neural network approach to minimize total completion time on a single batch processing machine. *International Transactions in Operational Research*, 0(0), 2019.

[SAB88] Subhash C. Sarin, Seokyoo Ahn, and Albert B. Bishop. An improved branching scheme for the branch and bound procedure of scheduling n jobs on m parallel machines to minimize total weighted flowtime. *International Journal of Production Research*, 26(7):1183–1191, 1988.

[SB99] I. Sabuncuoglu and M. Bayiz. Job shop scheduling with beam search. *European Journal of Operational Research*, 118(2):390–412, 1999.

[SCAC+19] Inés Sittón-Candanedo, Ricardo S. Alonso, Juan M. Corchado, Sara Rodríguez-González, and Roberto Casado-Vara. A review of edge computing reference architectures and a new global edge proposal. *Future Generation Computer Systems*, 99:278–294, 2019.

[Sch17] Klaus Schwab. *The fourth industrial revolution*. Currency, 2017.

[Sci] Towards Data Science. Applied deep learning - part 1: Artificial neural networks.

[SCP16] R. Mellado Silva, C. Cubillos, and D. Cabrera Paniagua. A constructive heuristic for solving the job-shop scheduling problem. *IEEE Latin America Transactions*, 14(6):2758–2763, 2016.

[SF96] Norman Sadeh and Mark S. Fox. Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem. *Artificial Intelligence*, 86(1):1–41, 1996.

[SFO86] Stephen Smith, Mark Fox, and P. S. Ow. *Constructing and Maintaining Detailed Production Plans : Investigations into the Development of Knowledge-based Factory Scheduling Systems*. 1986.

[SjLfBh07] W. Shi-jin, X. Li-feng, and Z. Bing-hai. Filtered-beam-search-based algorithm for dynamic rescheduling in fms. *Robotics and Computer-Integrated Manufacturing*, 23(4):457–468, 2007.

[SKK19] A. Safiullin, L. Krasnyuk, and Z. Kapelyuk. Integration of industry 4.0 technologies for "smart cities" development. In *IOP Conference Series: Materials Science and Engineering*, volume 497, 2019.

[SM98] Anant Singh and Sheik Meeran. *A State-Of-The-Art Review Of Job-Shop Scheduling Techniques*. 1998.

[SMF07] Mohammad Saidi-Mehrabad and Parviz Fattahi. Flexible job shop scheduling with tabu search algorithms. *The International Journal of Advanced Manufacturing Technology*, 32(5):563–570, 2007.

[sol] solwaycomms. What is edge computing and why should i care?

[SPKG18] Christos Stergiou, Kostas E. Psannis, Byung-Gyu Kim, and Brij Gupta. Secure integration of iot and cloud computing. *Future Generation Computer Systems*, 78:964–975, 2018.

[STW99] Yuri Sotskov, Thomas Tautenhahn, and Frank Werner. *On the Application of Insertion Techniques for Job Shop Problems with Setup Times*, volume 33. 1999.

[vis] visiontechme. Why move to cloud ?

[vLA87] Peter J. M. van Laarhoven and Emile H. L. Aarts. *Simulated annealing*, pages 7–15. Springer Netherlands, Dordrecht, 1987.

[Wan10] S. Wang. Filtered beam search based flexible job shop scheduling problem with transportation time. In *Advanced Materials Research*, volume 97-101, pages 2440–2443. 2010.

[WCW11] Hui-Mei Wang, Fuh-Der Chou, and Ful-Chiang Wu. A simulated annealing for hybrid flow shop scheduling with multiprocessor tasks to minimize makespan. *The International Journal of Advanced Manufacturing Technology*, 53(5):761–776, 2011.

[WcXx09] Z. Wu-cheng and L. Xin-xing. A negotiation model of supply chain based on multi-agent. In *2009 Second International Conference on Information and Computing Science*, volume 1, pages 228–231, 2009.

[WGK08] Gary R. Weckman, Chandrasekhar V. Ganduri, and David A. Koonce. A neural network job-shop scheduler. *Journal of Intelligent Manufacturing*, 19(2):191–201, 2008.

[Wor15] Design World. Big future for cyber-physical manufacturing systems, 2015.

[WR94] T. M. Willems and J. E. Rooda. Neural networks for job-shop scheduling. *Control Engineering Practice*, 2(1):31–39, 1994.

[WW95] Frank Werner and Andreas Winkler. Insertion techniques for the heuristic solution of the job shop problem. *Discrete Applied Mathematics*, 58(2):191–211, 1995.

[WYT15] Y. Wang, J. Yan, and M. M. Tseng. An auction based negotiation protocol for resource allocation in customized housing construction. *Procedia CIRP*, 28:161–166, 2015.

[WZX08] S. J. Wang, B. H. Zhou, and L. F. Xi. A filtered-beam-search-based heuristic algorithm for flexible job-shop scheduling problem. *International Journal of Production Research*, 46(11):3027–3058, 2008.

[XCW+10] Li-Ning Xing, Ying-Wu Chen, Peng Wang, Qing-Song Zhao, and Jian Xiong. A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Appl. Soft Comput.*, 10(3):888–896, 2010.

[XK15] A. S. Xanthopoulos and D. E. Koulouriotis. Cluster analysis and neural network-based metamodeling of priority rules for dynamic sequencing. *J. Intell. Manuf.*, 29(1):69–91, 2015.

[Xu12] Xun Xu. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(1):75–86, 2012.

[XZW15] Hongquan Xue, Peng Zhang, and Shengmin Wei. Applying a hybrid algorithm of immunity and ant colony in job-shop scheduling. In *Industrial engineering and manufacturing technology: Proceedings of the 2014 international conference on industrial engineering and manufacturing technology (ICIEMT 2014)*, volume 4, page 91, 2015.

[YLH+18] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang. A survey on the edge computing for the internet of things. *IEEE Access*, 6:6900–6919, 2018.

[YN96] Takeshi Yamada and Ryohei Nakano. *Job-Shop Scheduling by Simulated Annealing Combined with Deterministic Local Search*, pages 237–248. Boston, MA, 1996.

[YRN94] T. Yamada, B. E. Rosen, and R. Nakano. A simulated annealing approach to job shop scheduling using critical block transition operators. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 7, pages 4687–4692 vol.7, 1994.

[YZ17] D. Yang and X. Zhang. A hybrid approach for due date assignment in a dynamic job shop. In *2017 9th International Conference on Modelling, Identification and Control (ICMIC)*, pages 793–798, 2017.

[ZABA15] M. H. Zahmani, B. Atmani, A. Bekrar, and N. Aissani. Multiple priority dispatching rules for the job shop scheduling problem. In *2015 3rd International Conference on Control, Engineering & Information Technology (CEIT)*, pages 1–6, 2015.

[ZCBO91] D. N. Zhou, V. Cherkassky, T. R. Baldwin, and D. E. Olson. A neural network approach to job-shop scheduling. *IEEE Transactions on Neural Networks*, 2(1):175–179, 1991.

[ZDZ+19] Jian Zhang, Guofu Ding, Yisheng Zou, Shengfeng Qin, and Jianlin Fu. Review of job shop scheduling research and its new perspectives under industry 4.0. *Journal of Intelligent Manufacturing*, 30(4):1809–1830, 2019.

[ZGCG15] Boxuan Zhao, Jianmin Gao, Kun Chen, and Ke Guo. *Two-generation Pareto ant colony algorithm for multi-objective job shop scheduling problem with alternative process plans and unrelated parallel machines*. 2015.

[ZMM14] Qiao Zhang, Hervé Manier, and Marie-Ange Manier. A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints. *International Journal of Production Research*, 52(4):985–1002, 2014.

[ZP16] B. H. Zhou and T. Peng. Modified shifting bottleneck heuristic for scheduling problems of large-scale job shops. *Journal of Donghua University (English Edition)*, 33(6):882–886, 2016.

[ZR18] T. K. Zubaran and M. R. P. Ritt. An effective tabu search for job shop scheduling with parallel machines. In *IEEE International Conference on Automation Science and Engineering*, volume 2018-August, pages 913–919, 2018.

[ZW10] Rui Zhang and Cheng Wu. A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing*, 10(1):79–89, 2010.