FEDERAL UNIVERSITY OF MINAS GERAIS
SCHOOL OF ENGINEERING
DEPARTMENT OF MANUFACTURING ENGINEERING

Gabriela Naves Maschietto

Scheduling problem in a distribution center with two cranes subject to non-interference constraints

Belo Horizonte
2015

Gabriela Naves Maschietto

Scheduling problem in a distribution center with two cranes subject to non-interference constraints

Project Report submitted to the Graduate Program in fulfillment of the requirements of a Master degree in Manufacturing Engineering of the Federal University of Minas Gerais.

Supervisor: Prof. Dr. Martín Gomez Ravetti

Belo Horizonte
2015

Naves Maschietto, Gabriela.

Scheduling problem on two machines

99 pages

Dissertation (Master) - School of Engineering, Federal University of Minas Gerais. Belo Horizonte. Department of Manufacturing Engineering.

1. Scheduling problems
2. Parallel machines
3. Multiprocessors
4. Non-interference constraints
5. Genetic algorithm

I. Federal University of Minas Gerais. School of Engineering. Department of Manufacturing Engineering.

Abstract

This work is motivated by the economic impact of scheduling problems on a company's supply chain and by its applicability on the industrial and service environments. It addresses jobs sequencing on two cranes subject to non-interference constraints, while considering different modeling perspectives and storage policies. The problem is based on a real case at a distribution center of steel coils, where two cranes sharing the same rail must load a sequence of trucks, which have a defined demand of coils. A distribution center is taken as a scenario due to its logistic importance for companies from different sectors and due to the lake of research works in this field. This dissertation evaluates two types of parallel machine problems and one type of multiprocessors problem. And finally, two genetic algorithms are developed in order to find a good feasible solutions for the parallel machine cases.

Keywords: Mathematical Approaches for Scheduling, Warehouse Management Systems, Cranes Scheduling, non-Interference Constraints, Genetic Algorithm.

Resumo

Este trabalho é motivado pelo impacto econômico de problemas de sequenciamento na cadeia de suprimento e pela sua aplicabilidade em ambientes industriais e de serviços. Este estudo trata do sequenciamento de tarefas em dois guindastes sujeitos a restrições de não interferência, enquanto considera diferentes abordagens de modelagem e de políticas de estocagem. O problema é baseado em um caso real de um centro de distribuição de bobinas de aço, onde duas pontes, que compartilham o mesmo trilho, devem carregar uma sequência de caminhões. Esses por sua vez, têm uma demanda predefinida de bobinas. Um centro de distribuição foi tomado como base devido à sua importância logística para empresas de diferentes setores, assim como devido à falta de pesquisas nesta área. Esse trabalho avalia dois tipos de problemáticas de máquinas paralelas e um problema de multiprocessadores. Finalmente, dois algoritmos genéticos são desenvolvidos para encontrar boas soluções viáveis para os problemas de máquinas paralelas.

Palavras chaves: Abordagens Matemáticas para o Problema de Sequenciamento, Sistemas de Gestão de Armazéns, Sequenciamento de Guindastes, Restrições de não interferência, Algoritmo Genético.

Acknowledgments

I acknowledge my advisor, Prof. Martín Gómez Ravetti, for sharing with me his knowledge and lead me in the direction of personal and professional development.

To Prof. Farouk Yalaoui and LOSI laboratory for the exchange opportunity and precious advices.

To Prof. Maurício Cardoso de Souza and Prof. Yassine Ouazene for their helpful comments.

To my family, especially my parents and my sister, for the support and encouragement throughout my academic career.

To my friends, especially Babi, for the advices and friendship and to Vince for the inspiration.

And finally, I acknowledge CNPQ for the financial support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The companies have sought to gain competitive advantages over their competitors such as streamlining costs and improving service level. Optimization processes play a crucial role in the effective management of the Supply Chain, justifying the development of studies in this field.

The economic impact on different information processing, industrial and service environments motivates our interest in scheduling problems. Studies on this topic deal with the allocation of limited resources to tasks in a given time period, aiming to optimize some criteria. Such problems are present in manufacturing companies, as well as in other sectors, since the beginning of the first industries and have been quite explored by the operational research since the 1950s.

A distribution center is chosen as a scenario for analysis due to its logistic importance for companies from different sectors. It is important to highlight that the movement and handling of materials require resources that do not add value to the final product. The studied scenario considers the process of loading steel coils of a steelmaker's distribution center. According to Zäpfel and Wasner (2006), although the sequencing of the warehouse activities is considered critical to the supply chain competitiveness of steel, this problem has not yet received academic attention.

The scenario covers the sequencing of trucks'loading processes within the shed of the distribution center. The demanded load of each truck is defined by the lot sizing model of da Silva Neto (2013), who studied the same scenario, and whose instances were generated from real data. Each demanded load defines how many and which steel coils each truck should carry. Each coil is unique and has a known location in the shed.

From the foregoing, this study analyzes the cranes scheduling problem on a steelmaker's distribution center by different modeling perspectives. As the same scenario, can be understood as different environments of scheduling problems, in which exists several mathematical formulations with different computational complexity and difficulty of implementation.

Similarly, it is defined the arrangement policy of steel coils in the shed. This allows us to evaluate the impact of different forms of storing products while considering different machine environments problems. Thus, one is able to define the benefits of using each mathematical configuration, select and improve the best problem.

It is important to note that the trucks loading process is done by two cranes whose displacement are subject to non-interference constraints. It happens because both machines move on the same track inside the shed. According to Xie et al. (2014) there are few studies about cranes scheduling on steel warehouses. Few of them aim to sequence the loading and unloading of coils by single cranes, without considering non-interference constraints. The interference between machinery often appears at logistic centers, such as warehouses, stockyards and depots. They are especially found at port terminals, on both loading and unloading of ships, as on the handling area of the stockyards.

This study evaluates two modeling paradigms with different understanding of the studied environment. The first paradigm considers the parallel machines environment in which two problems are proposed. The second one considers a multiprocessors setting, in which two machines simultaneously process a single job; this is a variation of the parallel machines problem.

Mathematical models are proposed for the paradigms, in order to represent them properly. The problems of the parallel machine paradigm present the best performance, as indicated by the evaluation of the results. Because of that, this paradigm is taken as basis for the development of the next analysis. The mathematical models are not able to optimally solve real size instances due to the high computational time. Therefore, we decided to heuristically solve the selected problems.

Thereby, genetic algorithms are set to find good solutions, without guaranteeing the optimality, within a reduced computational time. Genetic algorithms belong to the class of evolutionary algorithms, which are inspired by models of the natural species evolution. They allow more direct interaction with the population of candidate solutions, to select suitable candidates to find the best solution.

1.1 Goals

This study aims to model and solve the scheduling problem in a two machines environment of a steel coils distribution center. Each machine corresponds to a crane, each job to a truck and each item to a coil. The mathematical models must consider the processing time of each job, which varies by machine and job. Intending to determine the best loading sequencing for each coil storage policy, one must analyze the best representation of the addressed scenario. One paradigm is selected to have its problems heuristically solved by a genetic algorithm. The paradigms considered in this work are: multiprocessors and unrelated parallel machines.

Seeking to meet the overall objective, the following specific objectives are outlined: i) contextualization of the studied scenario to define the problem; ii) development, implementation and selection of mathematical models able to represent the scenario and solve the truck scheduling problem subject to non-interference constraints. iii) Development and implementation of the genetic algorithms; iv) evaluation of the relationship between the modeling problems and the coils storage policies; and finally v) the literature review.

1.2 Published works

As a result of this dissertation, we have published two articles in two international conferences, (Maschietto et al., 2014) and (Maschietto et al., 2015). The first work presents the results obtained from the comparison between the mathematical model results of parallel machines problem and multiprocessors problem. The second one presents the results of two formulation perspectives of the parallel machine problem subject to non-interference constraints: using time-index and precedence index variables. As a final output, we expect to submit a journal article in 2015 integrating the results achieved in this work.

1.3 Organization of the dissertation

This work is divided into six chapters. The first one introduces the study, describes and justifies which are the expected goals.

The second chapter presents the based scenario and the operational characteristics of the distribution center of this study, as well as the focus area. It also describes the machine environments that will be addressed in this work.

Chapter 3 is devoted to the bibliography review about scheduling problems, weighted completion time optimization criterion, parallel machines, multiprocessor problems and non-interference constraints. As well as the review about computational complexities hierarchy evaluation and genetic algorithm heuristics.

Chapter 4 presents the considerations and formulations of the proposed paradigms. It also presents the instances definitions, results and analysis of the mathematical models performance with different instances configurations and steel coils arrangement policies.

Finally, in chapter 5 is presented the genetic algorithms, the parameters definitions and evaluation of their results. Chapter 6 concludes this dissertation and presents further works.

Chapter 2

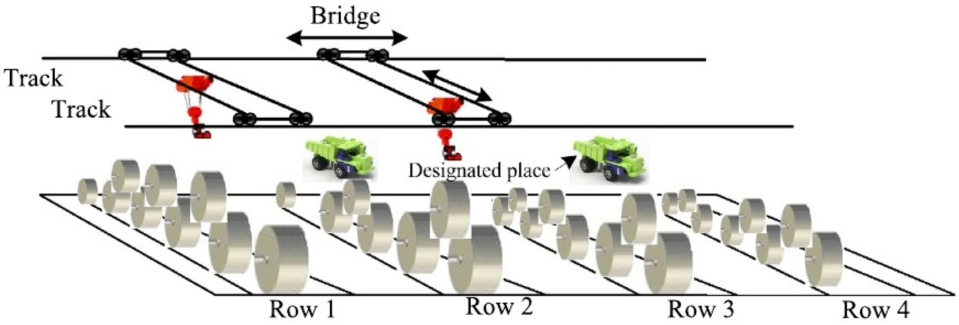
Context of the study

This chapter contextualizes the studied scenario which addresses the cranes scheduling problem in a steel coils distribution center, while considering different policies of products arrangement in the shed. First of all, it will be presented the distribution center characteristics and its operations, then the target problem and the machines configurations will be defined.

2.1 The distribution center scenario

The scenario of the proposed work is based on the distribution center (DC) operation of a steelmaker company. According to the Brazil Steel Institute (*Instituto Aço Brasil*), IAB (2013), this sector currently has 29 plants in Brazil, which produce 34.5 million tonnes of crude steel every year. The based steelmaker produces steel coils and steel sheets of different compositions, thicknesses and weights, which are mainly delivered to the automotive industry, construction and consumer goods sectors.

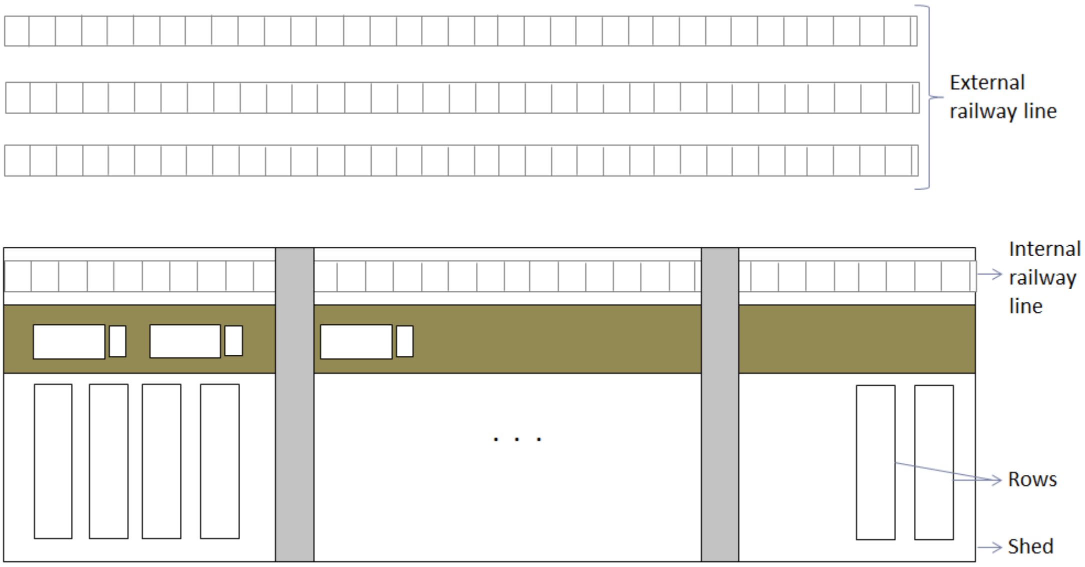
The products are sent to the distribution center via railroad and then they are sent to customers via road transportation. All the stock's material movements are made by cranes. A representation of the distribution center's warehouse can be seen in Figure 2.1. The picture shows the cranes, which move through the same track inside the shed and therefore are subject to displacement interference. It shows the storage rows of the steel coils, as well as the area where the trucks must go through for loading.



Xie et al. (2014), p.p. 2875, adapted.

Figure 2.1: Storage shed of steel coils

Basically, the whole physical structure of the distribution center is composed by a storage shed and an administrative building. There is a staging area to receive the truck before the loading process and another one to release it. There is also an outdoor area, with three railway lines, where the arrived loaded wagons are held until the products are unloaded. The Figure 2.2 illustrates the distribution center's shed and the external railway lines.



da Silva Neto (2013), p.p. 31, adapted.

Figure 2.2: The steel coils storage shed and the railway lines

2.1.1 Operations of the distribution center

At the distribution center, the products must pass through the steps of receiving, handling, storing, picking and shipping, which are detailed below.

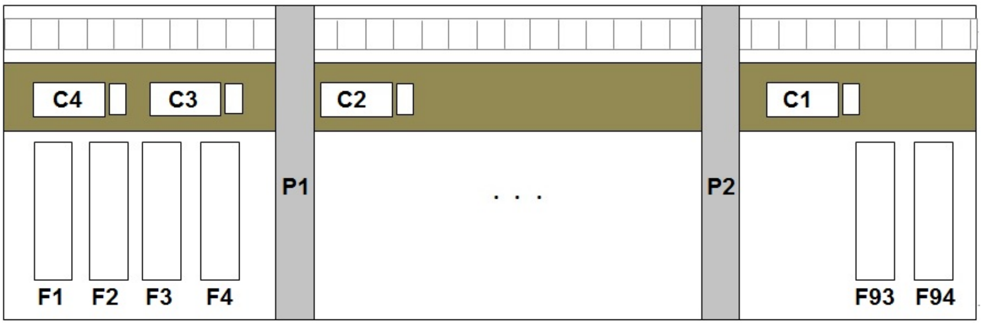
(a) Receiving: the wagons arrive from the steel mill with products and stand outside to wait the unloading process.

(b) Handling: the unloading process usually follows the arrival order of the trains, according to the FIFO rule (First In First Out). Each wagon is discharged by one of the two cranes at the night time, this operation do not interfere the loading's worktime.

(c) Storing: the materials are stacked into rows, which are indexed and separated by the following criteria: type (steel coil or sheet), size, destination (depending on the volume), client (depending on the volume) and weight. The last criterion sets the level at which the coil should be placed on the row. Heavier coils can not stay on the top of lighter coils due to issues of product quality and safety. Each coil has its row recorded for future location, but its exactly position into the row is not registered.

(d) Picking: this process considers the dispatch report, whose lots are defined by customers for delivery over the next 24 hours. During the process an operator must also scheduled the order's products into the designated truck at a specific time. The products must be checked and loaded into the trucks. One of the cranes is designed to retrieve the coils, depending on the coil location inside the warehouse and the machines availability. The truck loading can be done by one of the cranes, or by both of them, as can be seen at the example shown on Figure 2.3. The coil processing time varies according to its row position, which increases with the number of movements.

(e) Shipping: the truck passes through the steps of shipping preparation and release of material invoice.



da Silva Neto (2013), p.p. 31, adapted.

Figure 2.3: Loading example

Where C indicates truck, B steel coil and P crane. The numbers that follow C and P are, respectively, the truck and the crane identifier. The number that follows B is the steel coil's

Table 2.1: Loading example of the Figure 2.3

Truck	Number of coils	Rows of the coils	Used crane
C1	4	B45 — B50 — B80 — B93	P2 — P2 — P2 — P2
C2	3	B5 — B16 — B14	P1 — P1 — P2
C3	2	B2 — B90	P1 — P2
C4	2	B4 — B10	P1 — P1

elaborated by the author.

row.

It should be highlighted that the movement of material absorbs resources that do not add value to the product, hence the minimization of the handling implicates avoiding unnecessary movement and reduces the risk of damage or loss of product.

2.2 Analyzed scenario

The main scenario addressed in this study covers the picking stage with focus on the loading of trucks. Intending to get the best use of the distribution center, it shall be defined the policies of arrangement of steel coils and the machine environments to represent the trucks loading.

This study assesses different approaches of coils storage inside the shed and models the scenario as different machines environments to enable the evaluation of the configurations' benefits. Since cranes are expensive equipments, an effective solution to the scheduling problem should increase their utilization rate and provide better logistical support for the steel manufacturing process.

This work looks at the planning of coils, as they represent more than 90% of the materials in stock. Thus, the proposed models assume that each coil has a known location into the shed and each truck has a defined load demand, such as the specifications of how many and which coils should be loaded. The load demand of each truck is a parameter given by the lot sizing problem of da Silva Neto (2013), who studied the cargo assignment considering the same scenario.

This dissertation also assumes that the coil processing time is variable and depends on the coil location (row) on the shed, but it independs of the coil row level. The row level of the coils is disregarded because such information is nonexistent in the considered distribution center. However, once this data is available, the model could be adapted, and works as Zäpfel and Wasner (2006), Tang et al. (2014) and Xie et al. (2014), may be useful to underlie the study.

The analyzed scenario considers the interference between cranes, where one crane bounds the operational area of the other one, since both move through the same track inside the shed, see Figure 2.1. Interference between machinery often appears at logistic centers, such as warehouses, stockyards and depots. They are especially found at port terminals on both loading and unloading of ships, and on the material handling at stock yards.

Extensions can be identified from the analysed scenario, as (i) the consideration of more machines; or (ii) the assumption of row level of coils; (iii) the integration between arrivals and deliveries of steel coils; or (iv) the integration between the operations of steel coils and steel sheets. Moreover, future studies can focus on other areas of distribution centers, or on other steel production process. Likewise, other perspectives of the operational research can be addressed in the distribution center's scenario, such as lot sizing.

2.2.1 Machine environments

Before presenting the machine configurations; it is important to define the optimization criterion. As in the real case several clients have higher priorities, we opt to work with the minimization of the total weighted completion time. With this objective function we are able to distribute the work among the machines weighting at the same time jobs to prioritize its completion.

Seeking to find the best representation of the chosen scenario, two modeling paradigms are proposed, Figure 2.4. The first one considers a parallel machine environment and the second one a multiprocessor setting. Paradigm P1 defines two parallel machine problems, in which $MP1$ considers only the trucks scheduling and $MP2$ considers the trucks and coils scheduling. Paradigm P2 addresses a multiprocessors configuration, in which coils and trucks have to be sequenced. A paradigm P3 could also be defined as a flow shop machine environment. These machine configurations are detailed below.

I. Paradigm P1: Parallel machines

The problem can be modeled as a parallel machines if each truck must be processed by a single crane. This paradigm includes the interference between cranes during the removal of coils, as both machines move through the same track. P1 is subdivided into two problems: MP1 considers

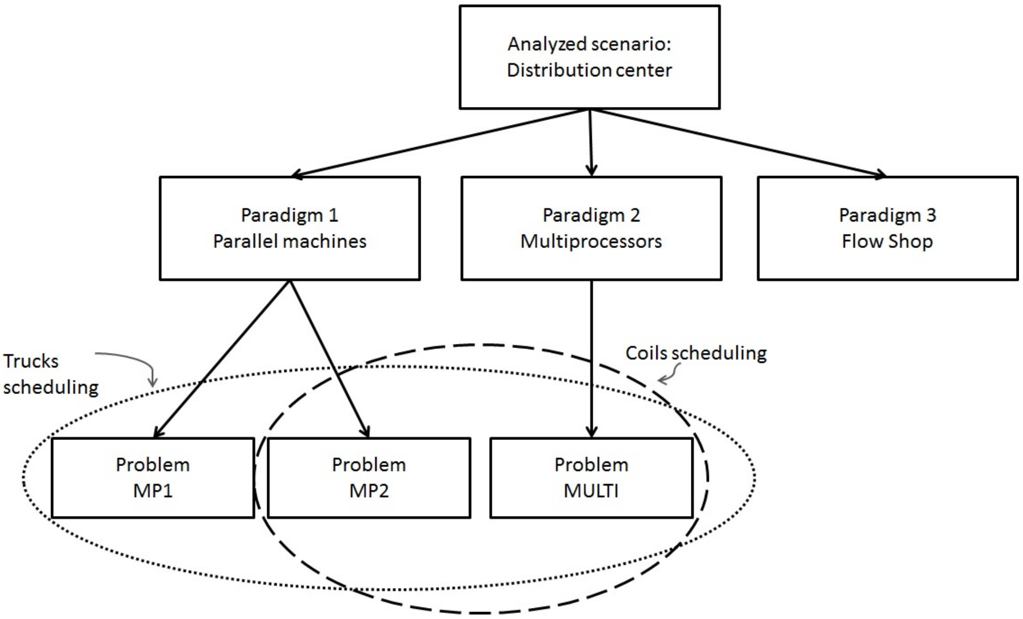


Figure 2.4: Machines configuration definition

truck interference, that is, if the cargo of two trucks contains incompatible coils, the trucks can not be processed at the same time. MP2 considers that the coils it-selves need to be scheduled, thus there is coils' interference.

In the $MP1$ problem, when truck j starts, it prevents other trucks $f \neq j$ to be processed if f has coils in some row between the area bounded by the rows of the coils from j.

In $MP2$ problem, the processing of a coil blocks the crane's operational area corresponding to the coil's storing position. Considering the example shown in Figure 2.3, one can sequence the jobs processing through $MP1$, Figure 2.5 (a), and through $MP2$, Figure 2.5 (b):

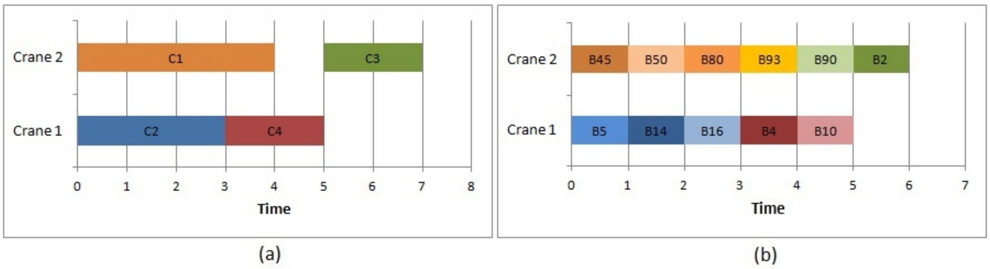


Figure 2.5: Sequencing of trucks by MP1 and MP2 problems, respectively

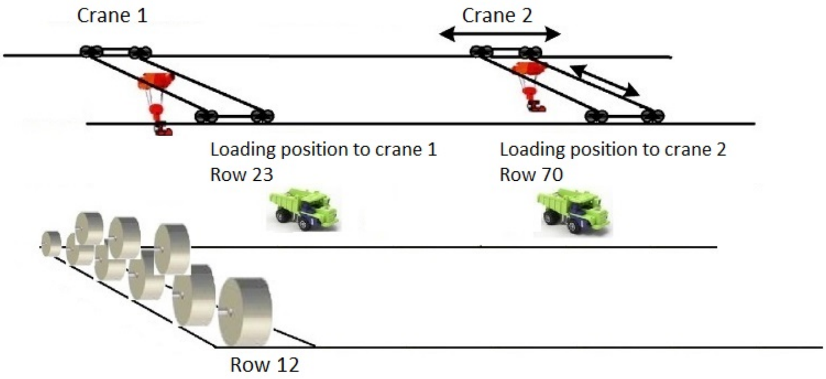
Table 2.2: Legend to Figure 2.5

C_j: Truck j	B_f: coil in the row f
C1	B45 — B50 — B80 — B93
C2	B5 — B16 — B14
C3	B90 — B2
C4	B4 — B10

As can be seen on Figures 2.5, (a) and (b), the coils of the same truck are made without preemption. The processing time of each coil is considered equal to one unit of time to facilitate the understanding.

As observed in Figure 2.5 (a), truck $C2$ has coils in rows 5, 14 and 16. So, when $C2$ starts processing, any other truck which has coils between the rows 5 and 16 can be processed simultaneously. Thus, only $C1$ can be loaded by crane 2 while $C2$ is processed by crane 1, because its operational area, row 45 to 93, does not match with the one of $C2$. Because of that, crane 2 can operate $C1$ without colliding with crane 1.

Furthermore, although these cranes are considered identical (both have the same technology), the time spent to remove a coil depends on the crane. It happens because there is a displacement time performed by the crane to move from the truck to the coil's row. Figure 2.6 shows an example of the removal of a coil stored in row 12. It is possible to see that the machine 2 takes longer to process this coil than machine 1, since the truck loading position depends on the machine.



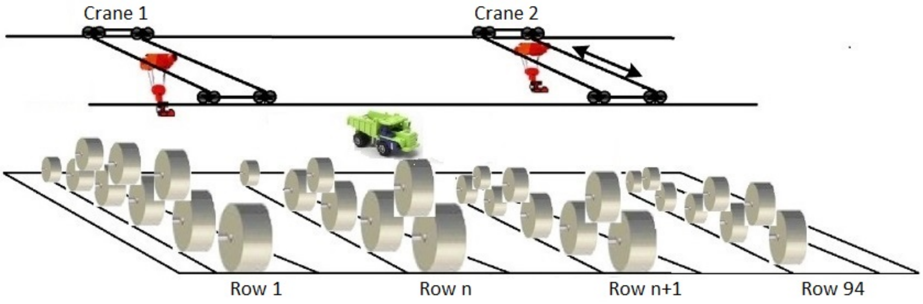
Xie et al. (2014), p.p. 2875, adapted .

Figure 2.6: Example of the displacement time spent by each crane to load the same coil

II. Paradigm P2: Multiprocessors

The studied scenario can be modeled as a multiprocessors problem, in which the two cranes can process different coils from the same truck simultaneously, Figure 2.7. In this paradigm, the cranes are treated as identical machines, because of their need to load the same truck, which is always in an established loading position, so that the distance between the vehicle and a coil is independent of the machine. Furthermore, it is considered the *nonfix* feature, since trucks with only one coil must be processed by one of the cranes, but it is not specified which one;

and trucks with two or more coils must be processed by the two bridges together (details will be given further on this work, at Chapter 3). This approach is called $MULTI$.

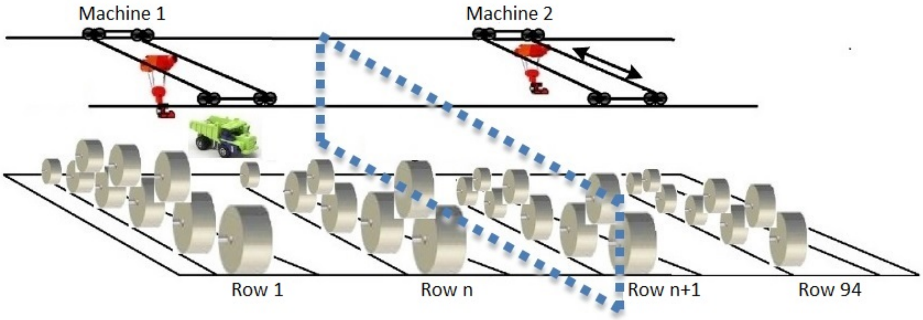


Xie et al. (2014), p.p. 2875, adapted .

Figure 2.7: The truck processing by the multiprocessors configuration

III. Paradigm P3: Flow shop

Another configuration, identified and contemplated by the work of da Silva Neto (2013), not included at this dissertation, is the flow shop environment. The scenario can be formulated as flow shop problem if the storage shed is divided into two parts, illustrated by the blue cut on Figure 2.8, where each machine is responsible for each subdivision. Thus, the trucks would enter the shed to be processed firstly by the crane 1 with the coils that are in the first subdivision, and finaly on the second one. In this case, each cranes is responsible for the coils stocked in their corresponding section.



Xie et al. (2014), p.p. 2875, adapted .

Figure 2.8: The truck processing by the flow shop approach

Chapter 3

Literature review

This chapter reviews the literature that guided this study. It focuses on scheduling problems with non-interference constraints and those minimizing the weighted completion time with parallel machine and multiprocessors settings. Moreover, it summarizes the differential of this study in relation to literature, reviews the complexity of these problems and concludes with the literature review on genetic algorithms.

3.1 Scheduling problems

The scheduling problems can be defined as the allocation of resources to tasks in a given period of time, in order to optimize one or more goals. Such problems are present in manufacturing companies since the rise of the first industries in the mid-eighteenth century. At the beginnings it was only listed the start of an order processing or the deadline of an order delivery (Herrmann, 2006).

In the early twentieth century, when production systems became more complex and large, Frederick Taylor proposed the creation of a production planning office, which would justify the use of formal methods for scheduling. At about the same time Henry L. Gantt created production control tables, which allowed to visualize the scheduling and validate the production status (Herrmann, 2006). Later, the computerized programming would appear, and the first approaches to scheduling problems would be made in the mid-1950s (Graham et al., 1979) and (Allahverdi et al., 2008).

Nowadays, scheduling problems play a crucial role in the production planning of a successful company, since the requirements for production flexibility and costumer satisfaction is always increasing. Scheduling problems have been widely explored in the literature and can be applied in several areas and types of industrial environments, services and information processing. General references about scheduling problems include the books of Baker (1974), Conway et al. (1967) and Pinedo (2008). Readings such as Lenstra et al. (1977), Graham et al. (1979), Cheng and Sin (1990), Hall and Sriskandarajah (1996), Drozdowski (1996), Lee et al. (1997), Allahverdi et al. (1999), Gupta and Stafford Jr (2006), Allahverdi et al. (2008) and Bierwirth and Meisel (2010) allows us to understand some specific schedule issues that are currently addressed.

Lenstra et al. (1977) study the complexity of some scheduling problems, such as parallel machines, flow shop, permutation flow shop and job shop. Graham et al. (1979) interpret the computational complexities and review studies about optimization algorithms, enumeration and approximation algorithms for the same deterministic scheduling problems addressed by Lenstra et al. (1977), as well as to single machines and open shop problems.

Cheng and Sin (1990) survey the parallel machine problem literature since its beginnings until 1990s. Hall and Sriskandarajah (1996) review problems about blocking and no-waiting processes. Drozdowski (1996) and Lee et al. (1997) investigate problems about 1-job-on-r-machines=, called multiprocessor task scheduling. Lee et al. (1997) also study scheduling problems with availability constraints and present works on local search techniques.

Allahverdi et al. (1999) and Allahverdi et al. (2008) compile existing literature between 1960 and 2008 on scheduling problems with setups for distinct machine configurations. While Gupta and Stafford Jr (2006) review studies about flow shop since the development of Johnson's rule in 1954. Bierwirth and Meisel (2010) review scheduling problems in port environments.

To facilitate the understanding of this work, the main elements that make up scheduling problems are presented:

- Resources: these are goods or services with limited or unlimited availability. These can be machines, workforces, tools, etc.. The studied resources are the two cranes of the storage shed of the distribution center.

- Tasks: these are the operations that must be performed in the production process. Each task requires =a certain amount of time and/ or resources. The task in the addressed problem is the loading of steel coils.

- Job: it is the set of tasks or operations that must be executed in a sequence, it must represent the technological order of a product. For each task of a job there is an associated processing time. Thus, in the covered scenario the trucks are the jobs.

- Objective function: generally corresponds to the company's main performance objective, it can take different shapes and have one or more optimization criterion. The most common criterion is the minimization of the makespan (C_{max}), which is the maximum completion time of a job between all the jobs to be process. According to Pinedo (2008), this has considerable practical interest, as its minimization is somewhat equivalent to the maximum machine utilization. The objective function addressed in this study is to minimize the sum of the weighted completion time because on the based scenario, some customers have priority among others.

According to the classical literature, scheduling problems can be classified into single machine, parallel machines or serial machines, such as flow shop, flexible flow shop, job shop, flexible job shop and open shop.

This study focus on modeling and implementing a scheduling problem of a distribution center scenario with two machines. The configurations analyzed are: I. two unrelated parallel machines and II. multiprocessor in which two servers process the same job.

The notation of three fields of Graham et al. (1979) will be used, $\alpha|\beta|\gamma$, where α represents the configuration of machines or resources, and it may take only one feature; β represents the characteristics of the process and problem's constraints, and can take more than one feature at the same time; and γ represents the performance measure to be optimized, which may have only one feature.

Once the cranes move on the same track, their displacement may interfere on each other's operation since one can not overtake the other. In addition, some coils can not be loaded simultaneously when their rows are very close, because the bridges should keep a safety distance between them. This are the non-interference constraints which will be defined by itf, to be included in the β field when needed. Thus, the literary review of the main topics covered by this study will be presented in the following order: weighted completion time objective function, parallel machine problems, multiprocessors problems, problems with non-interference constraints, synthesis, computational complexities and genetic algorithms.

3.1.1 Minimizing $\sum w_j C_j$

The weighted shortest processing time (WSPT) rule can find optimal solutions for $1||\sum w_j C_j$ problem (Skutella and Woeginger, 1999) and (Pinedo, 2008). According to this rule, the jobs must be ordered in decreasing order of $\frac{w_j}{p_j}$. The computational time needed to sequence the jobs on the $1||\sum w_j C_j$ problem according to WSPT is the time required to sort the jobs according to the ratio of the two parameters.

The $1|prec|\sum w_j C_j$ problem can be solved in polynomial time if the problem presents chain precedence constraints, but the problem is strongly NP-hard if it has arbitrary precedence constraints (Pinedo, 2008). The $1|r_j, prmp|\sum C_j$ can be solved in polynomial time, but its weighted version is strongly NP-hard, as well as the $1|r_j|\sum C_j$ problem (Lenstra et al., 1977).

Belouadah et al. (1992) propose a new branch-and-bound (B&B) algorithm for the $1|r_j|\sum w_j C_j$ problem. Their B&B incorporates three special features that contribute to its efficiency: lower bounds based on the idea of job splitting; restriction of its search tree, by the release date adjustments incorporated into the branching rule; and their dominance theorem. The authors also present a greedy heuristic which uses a list scheduling procedure that gives priority to job j for which p_j/w_j is minimal. It uses sufficient conditions to generate optimal schedules.

According to the classical theory, the shortest processing time (SPT) rule can optimally solve the $Pm||\sum C_j$ problem. According to the SPT rule, the smallest job has to go on machine 1 at time zero, the second smallest one on machine 2, and so on. Then the $(m+1)th$ smallest job goes through machine 1, the $(m+2)th$ on machine 2, and so on. However, the SPT rule can not optimally solve some problems with the addition of other features in the β field, which can turn it into a NP-hard problem. The $Pm|sds{,}r_j|\sum C_j$ problem, where sds is the sequence-dependent setup time, for example, is known to be strongly NP-hard, since the single machine approach $1|r_i|\sum C_j$ is NP-hard in the strong sense.

Nessah et al. (2007) develop a B&B algorithm for the $Pm|sds,r_j|\sum C_j$ problem. It can efficiently solve problems up to 40 jobs in the case of 2, 3 and 5 machines. The authors prove conditions for local optimality which is used to develop the lower bound. Nessah et al. (2007) also develop a dominance of partial schedules theorem in which the nodes of the B&B tree are removed. Their lower bound is computed by relaxing the considered problem such that it turns into a single machine problem. Then the modified SPRT rule (shortest processing remaining time rule) is applied to optimally schedule the jobs.

If the parallel machine problem has the $\sum w_jC_j$ optimization criterion, the WSPT rule can not solve it optimally. Nonetheless, according to Pinedo (2008), it can provide a good approximation for the $Pm||\sum w_jC_j$ problem and its worst case analysis yields the lower bound $\frac{\sum w_jC_j(WSPT)}{\sum w_jC_j(OPT)} < \frac{1}{2}(1+\sqrt{2})$. According to Skutella and Woeginger (1999), as the $P2||\sum w_jC_j$ problem is NP-hard in the ordinary sense, it can be solved in pseudopolynomial time. When more than 2 machines are considered ($Pm||\sum w_jC_j$), the problem is NP-hard in the strong sense, excepting for the case of $w_j = 1$.

According to Li and Yang (2009), the majority of the researches about parallel machine problem with minimization of $\sum w_jC_j$ focus on studying the case of identical machines. Belouadah and Potts (1994), for example, propose a B&B with lower bounds based on Lagrangean relaxation for the $P||\sum w_jC_j$ problem. The formulation of their problem has unit time intervals and time index variables.

To compute the lower bound, Belouadah and Potts (1994) firstly schedule the jobs by a simple WSPT heuristic and then compute the multipliers of the Lagrangean relaxation. Some properties are defined to solve the relaxation, these properties must restrict the search for its optimal solution, which is a lower bound to the original problem. The authors prove that their bound is the optimal solution if the problem has one machine or $p_j = 1$ for all jobs j. They show some dominance rules and apply the WSPT heuristic at each node of the search tree to generate upper bounds.

Skutella and Woeginger (1999) give a PTAS (Polynomial Time Approximation Scheme) for the problem $P||\sum w_jC_j$, which is based on ratio-partitioning. The PTAS implies that for any given $\varepsilon > 0$ it will produce a solution within a *performance guarantee* or *performance ratio* of $(1 + \varepsilon)$ of the optimum value. The authors derive a PTAS for the case in which the largest job ratio is a constant factor away from the smallest job ratio (there are a bounded weight to length ratios of jobs). And they also present a similar result for the general problem. The idea is to find near optimal schedules for the subsets of jobs, which were partitioned according to their p_j/w_j ratios and then were geometrically rounded. These subsets can be later concatenated according to the WSPT rule.

In the same way Afrati et al. (1999) develop a PTAS for the $1/P/Rm|r_j|\sum w_jC_j$ problems. According to them, $R||\sum C_j$ can be solved by matching techniques, and the $Rm||\sum w_jC_j$ by PTAS. The authors perform several transformations that simplifies the input problem without dramatically increasing the objective value, such that the final result is amenable to a fast dynamic programming solution. They make a geometric rounding, which breaks time into geometrically increasing intervals; a time stretching, by adding small amounts of idle time spreaded throughout the schedule; and a weight-shifting, which moves the excess jobs to the next interval. Their small jobs are scheduled by WSPT rule at the identical parallel machine problem, but it does not hold for unrelated machines.

Pfund et al. (2004) survey the literature about solving traditional unrelated parallel machine scheduling problems, including the studies on minimization of the total weighted completion times. Although the problem $Rm||\sum C_j$ can be solved optimally in polynomial time, the $Rm||\sum w_jC_j$ is NP-hard (Pinedo, 2008).The authors show that an optimal schedule of this problem would have not any idle time within the schedule and the jobs must be ordered on each machine in WSPT order.

Li and Yang (2009) collect and classify models and relaxations, and survey some familiar heuristic algorithms and optimizing techniques for the $Q/R|\dots|\sum C_j/\sum w_jC_j$ problems. They review some exact methods for these problems in which some authors developed a B&B or a decomposition approach involving dynamic programming algorithms for solving the subproblem. The methods evaluated by them use the idea of WSPT in some of their parts. Li and Yang (2009) claim that there are few works about exact algorithms to the $Q/R|\dots|\sum w_jC_j/\sum C_j$ problems.

According to Li and Yang (2009), some general heuristics make use of WSPT rule directly or indirectly to obtain good solutions. They claim that there are few $Q/R|\dots|\sum w_jC_j/\sum C_j$ problems that are solved by local search techniques. According to them, the $Q|\dots|\sum C_j/\sum w_jC_j$ problems have not received much attention; there is no paper in which Lagrangean relaxation is used for $Q/R|\dots|\sum C_j/\sum w_jC_j$ problems; it is important to develop exact algorithms for $Q/R|\dots|\sum C_j/\sum w_jC_j$ problems; and there are few papers involving metaheuristics for the $R||\sum w_jC_j$ problem.

3.1.2 Parallel machines

These scenarios are important because the occurrence of resources in parallel is common in real world applications, being found in various industrial segments. Cheng and Sin (1990) review the studies about parallel machines since its earliest days. According to them, the studies in this area began in 1959 with the publication of the first article (McNaughton, 1959).

The parallel machines configuration can be found as identical or non-identical machines due to the need to produce products with some differences, or due to the level of the equipments technology. The parallel machine problems can be classified as:

(a) Identical machines: machines with the same processing speed. The jobs require only one operation and can be processed in any of the m machines, or any one of which belongs a particular subset.

(b) Uniform machines: parallel machines with different processing speeds and setups, such that $p_{ij} = p_j/s_i$ where p_j is the processing time of job j and s_i is the speed of machine i. It happens due to the characteristics of the jobs or the machine technology. This type of problem is common because the industries typically buy new equipments with advanced technology, but retains the old machinery.

(c) Unrelated machines: the processing time speed of each job is arbitrary and depends on the job and the machine where it is processed.

For problems with identical, uniform and unrelated parallel machines we add, respectively, Pm,Qm and Rm in the α field of the used notation, where m denotes the amount of considered machines.

Parallel machines scheduling problems should be treated as a two-stage process (Shim and Kim, 2007) and (Pinedo, 2008). It must be determined which jobs should be allocated on each machine and then sequencing them. The makespan minimization ensures a good load balance on the machines of the parallel machine problems (Pinedo, 2008).

Some authors, such as Yalaoui and Chu (2002) and Shim and Kim (2007), study the identical parallel machine problem with minimization of the total tardiness. They propose a B&B and present dominance properties of partial schedules to increase its speed. These properties allow the identification of jobs and partial scheduling that should be prioritized. Both papers present efficient algorithms.

Rocha et al. (2008) study the $Rm|sds|C_{max} + \sum w_j T_j$ problem. The authors present two mixed integer programming (MIP) models, based on Manne and Wagner models for job shop. They develop a B&B algorithm with better performance than the two models.

de Paula et al. (2010) study the $Rm|ST_{sd}, d_j| \sum w_j T_j$ problem and suggest a non-delayed relax-and-cut algorithm based on Lagrangean relaxation with a time-indexed formulation. A Lagrangean heuristic is also designed to obtain approximated solutions. The proposed methods generate optimal solutions within a feasible computational time, and obtain good optimality gaps for non-optimal solutions.

Other references on parallel machines include Pfund et al. (2004) and Kravchenko and Werner (2011), who survey, respectively, the literature of traditional scheduling problems about unrelated parallel machine and studies involving identical and uniform machines problems with the same processing times. Other reviews that also cover parallel machines include Lenstra et al. (1977), Graham et al. (1979), Allahverdi et al. (1999) and Allahverdi et al. (2008).

3.1.3 Multiprocessors

Classical scheduling problems consider that a machine handles only one job, and the processing of a job is made by only one machine, in a given time period. Unlike such problems, in multiprocessors problems a job must be processed simultaneously by r machines, where $r \in \mathbb{Z}$ (case addressed in this study), or many jobs can be processed by a single processor simultaneously $(0 < r \leq 1)$.

According to Lee et al. (1997), the problems of processing jobs by r machines simultaneously have gained importance from the 1980s, with the development of parallel computing systems.

These authors have interesting results showing that the increasing value of r generally increases the problem complexity. Lee et al. (1997) define two categories of multiprocessors: nonfix, or size, and fix. The former assumes that each job requires a fixed number of machines working simultaneously, but the required machines are not specified. The latter category fixates the set of machines for each job.

Basically, these scheduling problems are a variation of parallel machine problems, because they include one of both features on β field of the used notation. According to Lee and Cai (1999), the *nonfix* problem, for example, may be equivalent to the traditional parallel machine problem or the single machine problem, depending on the configuration of the *nonfix.* According to Lee et al. (1997), there is no clear relationship between the complexity of *nonfix* and *fix* models.

Drozdowski (1996) reviews studies about multiprocessors problem with parallel machines characteristics or dedicated processors characteristics. Similarly to the classical parallel machines theory, the multiprocessors can be classified as identical, uniform or unrelated machines.

An important study about jobs processed simultaneously by r machines can be found in Lee and Cai (1999). The authors study the $P2|nonfix_j|\sum w_jC_j$ and $P2|nonfix_j|L_{max}$ problems, in which the tasks must be performed by one or two processor simultaneously. They show the strong NP-hardness of these problems even when $w_j = w$. For the first problem, they establish optimality properties and find a dynamic programming algorithm to solve it optimally. A heuristic is developed as the computation time of the dynamic algorithm is relatively large. The authors also create an algorithm to find optimal solutions when all processing times (pj) are equal p.

Hoogeveen et al. (1994) investigate the computational complexity of multiprocessor scheduling problems with the *fix* characteristic. They evaluate the complexity of $Pm|fix_j|C_{max}$ and $Pm|fix_j|\ sumC_J$ problems, as well as their complexity when there are precedence constraints or jobs availability constraints. According to them, the introduction of release dates or precedence constraints usually makes the problem NP-hard. Furthermore, few attention has been done to the complexity of multiprocessors scheduling problems.

Lee et al. (1997) also distinguish the *sets* category, in which few attention had been given until the publication of their work. In the *set* type, one job can choose a set of alternative which has several dedicated machines. Consider, for example, a $m = 3$ machines problem and a job

j which can be processed by the set of alternatives $S_j = \{(M1, M2, M3), (M1, M2), (M2, M3), (M2)\}$. Both *fix* and *nonfix*, are special cases of *sets*, which is reviewed by Chen and Lee (1999).

Chen and Lee (1999) prove that the problem $Pm|set_j|C_{max}$ is NP-hard even for $m = 2$. A pseudopolynomial algorithm can solve it by assigning a set of alternatives for each job and sequencing them. The lower bound generated by the algorithm is the optimal solution of the $P2|set_j|C_{max}$ problem, when assuming any slack time on the machines. For a three machine problem, a fully polynomial algorithm in combination with a heuristic is elaborated to solve allocation and sequencing, respectively. Their results are extended to a general problem of m machines.

3.1.4 Non-interference constraints

According to Xie et al. (2014), there are few studies about crane scheduling in warehouses of steel coils, and there are even fewer about scheduling the shuffling operations of coils. The shuffling operation is the removal of coils that block the demanded coils which are in a lower level. According to the authors, researches about scheduling problems in warehouses are generally about sequencing the loading and unloading of coils on single machines, such as Zäpfel and Wasner (2006) and Tang et al. (2014), without considering the non-interference constraints or the interrelationship between transport and shuffling operations.

Zäpfel and Wasner (2006) study the crane scheduling problem, where the crane needs to store, remove and shuffle coils. The problem is treated as a job shop problem and is formulated as a nonlinear integer programming model. This model is difficult to solve, it has many restrictions and many variables, which are mutually dependent. Therefore, the authors propose an approximate method based on a local search algorithm.

Tang et al. (2014) study the single crane scheduling problem to minimize the makespan. The authors subdivide the warehouse such that a single machine must perform the pickup and shuffling operations into one of the subsets. Their problem should determine the coils removal sequence and the new positions for the blocking coils. They formulate a new mixed integer linear programming (MILP) model, prove the NP-hardness of the problem and propose an approximate algorithm to solve it. In this approach, the problem is divided into stages of shuffling and sequencing the loading of demanded coils. The coils exchange decision is taken at each stage with the support of the reduced MILP model. The authors propose a dynamic programming algorithm to optimally solve the restricted case and approximately solve a general case. They also present an heuristic to solve the general problem with good quality solutions.

Xie et al. (2014) have an interesting study about multiple cranes scheduling problem on a distribution center of steel coils. They know the exact position (row and level) of each coil and assume shuffling operations, once one coil can be blocked by other superiorly. Their problem, as well as in Zäpfel and Wasner (2006) and Tang et al. (2014), do not consider the coils weight related to their level, nor the non-preemption of the truck loading. The proposed model determines the position to send the blocking coils and sequences the transporting of the demanded coils and blocking coils. Xie et al. (2014) consider the interference on cranes displacement and the safety distance between them. As in Tang et al. (2014), each crane has retrieval time and placing time associated, as well as a travel speed along the track and movement speed along the support of the bridges. Such speeds assumes different values for loaded and unloaded displacements.

Xie et al. (2014) formulate the problem as a MILP model to minimize the makespan. They use positional variables and present some properties of viability and optimality in order to avoid interference between the cranes. As this problem is NP-hard, the authors propose a heuristic based on these properties and calculate the lower bound. The heuristic considers each type of operation (coil transportation or empty movement) as a scheduling problem. The performance of the heuristics is analyzed for the worst case, and computational experiments are generated for the MILP model and the heuristic with instances of 2 to 6 coils. The performance of the heuristic is evaluated for bigger instances. It solves all instances almost instantly and is able to generate good quality solutions.

Although references about scheduling problems with non-interference constraints are not commonly found for warehouses environments, this kind of problems have been widely discussed in port terminal environments. At this context, the problem can be treated as the materials exchange process between vessels and piers by quay cranes, which are specialized cranes that displace through rails next to the pier. Or as the loading process at the storage yards by yard cranes, which are container handling equipments which load and unload containers on trucks.

According to Kim and Park (2004) and Bierwirth and Meisel (2010), the quay cranes scheduling problem with non-interference constraints differs from the parallel machine problem due to precedence constraints of unloading and loading of vessels and due to the presence of interference on the displacement of the cranes. Quay crane studies include Bierwirth and Meisel (2010), Kim and Park (2004), Zhu and Lim (2006), Lim et al. (2007), Lee et al. (2008), Bierwirth and Meisel (2009) and Chung and Choy (2012). And Ng (2005) and Li et al. (2009) exemplify works about yard cranes.

A survey about berth allocation, quay crane allocation and quay crane scheduling problems at port terminals can be found in Bierwirth and Meisel (2010). The authors suggest a classification scheme for the addressed problems and conclude that the trend is the integration of these problems.

According to Bierwirth and Meisel (2009) and Bierwirth and Meisel (2010), makespan is the most common optimization criterion found in the literature about cranes scheduling. The authors claim that the non-interference constraints are present in the majority of the studies, although there are few when considering the safety distance of the cranes displacement. The crane attributes, such as travel speed and availability, are also quite neglected.

Non-interference constraints were first included in the quay crane scheduling problem model of Kim and Park (2004) who studied the scheduling of loading containers on a single ship, with two optimization criterion. They formulate a MIP model with linear ordering variables, propose a branch-and-bound and a GRASP heuristic (greedy randomized adaptive search procedure). They consider the location and release date of quay cranes as well as the presence of sets of tasks which can not be processed simultaneously.

Zhu and Lim (2006) formulate the $P2|itf|C_{max}$ as an integer programming problem with linear ordering variables. After prove its NP-hardness, they create a B & B algorithm to obtain the optimal solution and develop a simulated annealing metaheuristic to solve large problems.

Lim et al. (2007) develop a MIP model for the $Pm|itf|C_{max}$ problem. They consider each task as the whole compartment of certain vessel, and containers arranged in stock on ascending order of the compartments location, characterized into a unidirectional scheduling problem. The authors show that the problem can be decomposed: given the jobs allocation to cranes, the jobs sequencing will minimize the makespan. Thus, there will always be an optimal schedule among the unidirectional sequences. They also prove the complexity of the problem when $m \geq 2$ and develop a simulated annealing to solve larger instances.

Motivated by Kim and Park (2004), Lee et al. (2008) and Chung and Choy (2012) present a MIP model with precedence index variables and propose a genetic algorithm heuristic for the quay cranes scheduling problem with non-interference constraints. As in Kim and Park (2004), their models sequence only the loading of one ship, such that each quay crane must work on a single partition of the vessel until fulfill it. Lee et al. (2008) also prove the NP-hardness of the problem.

Bierwirth and Meisel (2009) present a revised optimization model for the $Pm|itf, prec|C_{max}$ problem and propose a heuristic in which a B&B is applied to exhaustively search the space of unidirectional schedules. This algorithm considers the non-interference constraints. The assignments of jobs to machines are generated through the B&B search tree and allow the unidirectional movement of cranes. A task sequence is determined by a disjunctive graph model for each crane such that an unidirectional schedule is given. The B&B of Bierwirth and Meisel (2009) does not find the tighter lower bounds but it optimally solves all instances or finds the best known solution quality. Their procedure allows to enumerate good schedules very fast.

Ng (2005) formulates the $Pm|itf, r_j|\sum C_j$ problem as an integer program with time index variables. A heuristic of two phases is proposed: firstly, the problem of m yard cranes is decomposed into m single machine problems, by partitioning the covered area of the cranes. Then a dynamic programming approach is used to determine an effective partition where a greedy heuristic is proposed to solve the yard cranes scheduling problem. The second phase reassigns the jobs to improve the schedule obtained on the first phase. The computational results show that the total completion time found by the heuristic is on average 7.3% above the lower bound developed to evaluate its performance.

Li et al. (2009) are the first to consider the safety distance between yard cranes, the simultaneous container storage/retrieval, and non-interference constraints, which was already addressed in previous studies. They develop a MIP model with few integer variables, due to the use of decision variables with only two indexes. They address three optimization criteria, develop heuristics and a rolling-horizon algorithm.

3.1.5 Synthesis of the literature reviews

The addressed problem of this dissertation is defined by two modeling paradigms: paradigm $P1$, defined by two unrelated parallel machine problems; and paradigm $P2$, represented by a multiprocessor problem with two identical machines. These problems consider the non-interference constraints and aim to minimize the total weighted completion time. As presented on this chapter, SPT rule can solve some problems with minimization of completion time, and its weighted version can be solved by WSPT rule. However these methods can not optimally solve our problems.

Furthermore, most researches on parallel machine problems with minimization of $\sum w_j C_j$ focus on identical machines (Li and Yang, 2009). This literature review shows that 89% of the addressed works on non-interference constraints consider the identical parallel machines. Moreover, according to Bierwirth and Meisel (2009) and Bierwirth and Meisel (2010), makespan is the most common optimization criterion on cranes scheduling problems. As observed in the literature review, 78% of the adressed studies on non-interference constraints consider the makespan, fully or partially, as the optimization criterion. As example of Kim and Park (2004), Bierwirth and Meisel (2009) and Xie et al. (2014).

Our research did not show works that address exactly the same crane scheduling problem subjected to non-interference constraints, but there are researches that address problems with some similarity. Few studies consider the crane scheduling problem for steel coils warehouses, such as Zäpfel and Wasner (2006), Tang et al. (2014) and Xie et al. (2014), and most of them deal with single crane scheduling. Differently than discussed in this dissertation, these works consider the row and level known of each coil and the presence of shuffling operations. Our dissertation does not consider shuffling because the coils level information is unknown in the proposed scenario.

Moreover, these three works do not consider that the trucks have defined coils demands and must be processed without preemption. The preemption feature is considered when, in quay cranes scheduling problems, trucks are assumed to be holds of vessels. It is also assumed in yard cranes problems, because every truck can only carry one container at a time.

Xie et al. (2014) study multiple cranes subject to non-interference on warehouse environments, but most of the researches on this constraints are into port terminal environment. Studies about quay cranes scheduling problems usually regard the interference into holds of ships, as example of Zhu and Lim (2006), Lee et al. (2008) and Bierwirth and Meisel (2009). Generally, they do not consider interference on storage field, nor the area bounded by the rows of hold's products, as addressed in $MP1$ problem of the parallel machine paradigm. On the other hand, Kim and Park (2004) treat the non interference by considering sets of tasks that can not be processed simultaneously.

The literature review also points out that these studies do not analyse the impact of different modeling configurations, nor the influence of products arrangement on the problems performance. The studied problems on port and warehouse environments do not consider scheduling of the combination of products, nor their vehicles, as addressed by $MP2$ and $MULTI$ problems. They generally only consider the holds or products scheduling.

According to Bierwirth and Meisel (2009) and Bierwirth and Meisel (2010), there are few studies considering the safety distance of cranes displacement. The crane attributes, such as travel speed and availability are also neglected. This project considers the safety distance between cranes and the travel time of the cranes is summed to the retrieval time of each coil. This can be assumed as the trucks are loaded in a position that depends on the crane and each travel time depends on the coils location (further explanation is on Chapter 4).

Kim and Park (2004), Bierwirth and Meisel (2009) and Chung and Choy (2012), for example, consider travel time separetly of the processing time, because the loading holds are not in a established position and the crane must move to each hold position to fulfill it. Lee et al. (2008) only consider the holds processing time.

The problems treated by the analysed literature are summarized in Table 3.1, as well as the related techniques to solve them. There are, in average, two *papers* regarding B&B, genetic algorithm, greedy heuristics, simulated annealing or developed their own heuristics.

Table 3.1: Problems and techniques covered by the analyzed articles

Work	**Problem**	**Technique**
Belouadah et al. (1992)	$1\|r_j\|\sum w_jC_j$	B&B
Belouadah and Potts (1994)	$P\|\|\sum w_jC_j$	B&B
Skutella and Woeginger (1999)	$P\|\|\sum w_jC_j$	PTAS
Afrati et al. (1999)	$1/P/Rm\|r_j\|\sum w_jC_j$	PTAS
Lee and Cai (1999)	$P2\|nonfix_j\|\sum w_jC_j$ $P2\|nonfix_j\|L_{max}$	dynamic programming and heuristic
Yalaoui and Chu (2002)	$Pm\|\|\sum T_j$	B&B
Kim and Park (2004)	$Pm\|itf\|C_{max} + \sum C_j$	B&B and GRASP
Ng (2005)	$Pm\|itf, r_j\|\sum C_j$	heuristic based in dynamic programming and greedy heuristic
Zhu and Lim (2006)	$P2\|itf\|C_{max}$	B&B and simulated annealing
Zäpfel and Wasner (2006)	$1\|\|C_{max}$	heuristic based on local search
Lim et al. (2007)	$Pm\|itf\|C_{max}$	simulated annealing
Shim and Kim (2007)	$Pm\|\|\sum T_j$	B&B
Nessah et al. (2007)	$Pm\|sds,r_j\|\sum C_j$	B&B
Rocha et al. (2008)	$Rm\|sds\|C_{max} + \sum w_jT_j$	B&B
Lee et al. (2008)	$Pm\|itf\|C_{max}$	genetic algorithm
Bierwirth and Meisel (2009)	$Pm\|itf, prec\|C_{max}$	heuristic with B&B
Li et al. (2009)	$1/Pm\|itf, d_j\|\sum E_m + \sum L_m$	rolling-horizon algorithm and heuristics
de Paula et al. (2010)	$Rm\|sds, d_j\|\sum w_jT_j$	relax-and-cut algorithm
Chung and Choy (2012)	$Pm\|itf, r_m\|C_{max} + \sum C_m$	genetic algorithm
Tang et al. (2014)	$1\|\|C_{max}$	dynamic programming and heuristics
Xie et al. (2014)	$Rm\|itf\|C_{max}$	greedy heuristic

3.1.6 Evaluation of the computational complexity hierarchy

To assess the complexity of the addressed paradigms it is considered the complexity hierarchy presented by Lenstra et al. (1977) and Pinedo (2008). In such hierarchy, a problem $P1$ may be reduced to $P2$ if, for any instance of $P1$, a instance of $P2$ can be constructed in limited polynomial time, such that solving the instance of $P2$ will solve the instance of $P1$. Thus, a procedure for a scheduling problem $P1$ can be applied to $P2$ if this is a special case of $P2$. Similarly, if $P1$ is NP-hard then $P2$ will also be NP-hard, but the reverse is not true. This relationship can be denoted by:

$$P1 \propto P2$$

Lenstra et al. (1977), Graham et al. (1979) and Pinedo (2008) show the relationship between different scheduling problems given the change in the elements of the problem classification. Figures 3.1 *a*, *b* and *c* show the elementary reductions between deterministic scheduling problems and indicate the consequences of the development of a new polynomial algorithm or proof of NP-hardness.

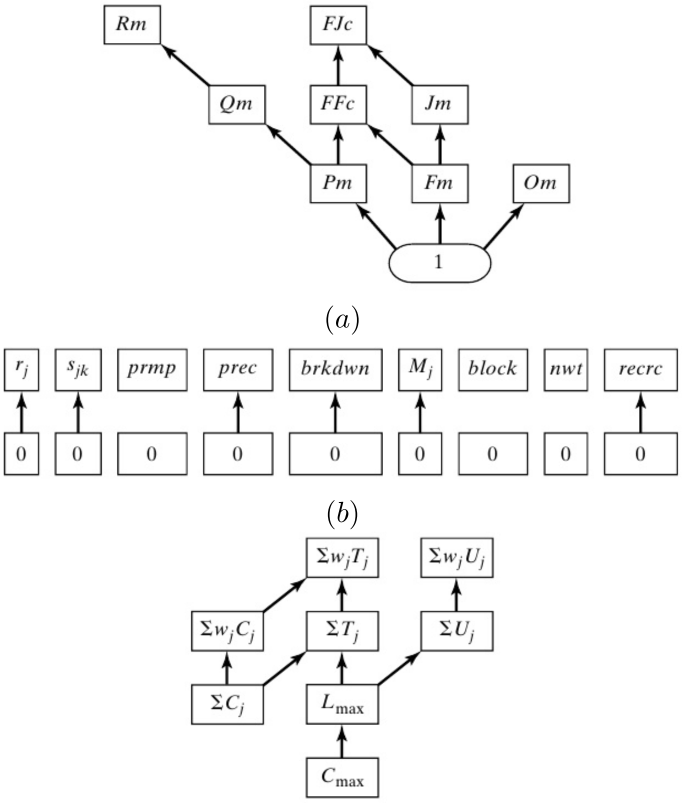
Defining the problem's complexity can justify the use of enumerative and heuristic methods. The yard cranes and quay cranes scheduling problems are NP-hard, according to, respectively, Ng (2005) and Lee et al. (2008). Ng (2005) further states that even if the problem is partitioned into m operational areas, in which each of the m machines is responsible for one of them, then each sub-problem is a NP-hard problem.

From the foregoing, this dissertation shows the complexity of the addressed paradigms.

I. Parallel machines paradigm

The problems $P2||C_{max}$, (Graham et al., 1979) and (Pinedo, 2008), and $P2||\sum w_jC_j$, (Graham et al., 1979) are NP-hard. Zhu and Lim (2006) show that the $P2|itf|C_{max}$ problem is strongly NP-hard, and Lim et al. (2007) prove the NP-hardness to problems of the same type but with

Figure 3.1: Complexity hierarchy of the fields α, β and γ, respectively



(a)

(b)

(c)

Pinedo (2008), p.p 27.

$m \geq 2$ machines.

Given the addressed problem, $R2|itf|\sum w_jC_j$, a special case can be derived by considering the same job processing time on each machine and the makespan as the optimization criterion. Thus, as the special case is strongly NP-hard, the $R2|itf|\ sumw_jC_j$ problem is also strongly NP-hard.

This can be verified by the following relation:

$$P2|itf|C_{max} \propto P2|itf|\sum w_jC_j \propto R2|itf|\sum w_jC_j$$

II. Multiprocessors paradigm

According to Drozdowski (1996), the problem $Pm|nonfix_j, p_j = 1|C_{max}$ is strongly NP-hard. Lee and Cai (1999) show that the problems $P2|nonfix_j|\sum C_J$, $P2|nonfix_j|\sum w_jC_j$ and $P2|nonfix_j|L_{max}$

are also strongly NP-hard, although $P2|nonfix_j, p_j = p|\sum w_jC_j$ can be solved in polynomial time.

The following relationship exists:

$$P2|nonfix_j|\sum w_jC_j \propto P2|nonfix_j, itf|\sum w_jC_j$$

Thus, given the multiprocessor approach studied in this dissertation, $P2|nonfix_j, itf|\sum w_jC_j$, one can consider the special case where the cranes can move freely in the shed. In this case, there is no interference and the problem is reduced to $P2|nonfix_j|\sum w_jC_j$. As this problem is strongly NP-hard, the one subject to non-interference constraints will also be.

3.2 Genetic algorithm

As the two adressed parallel machine problems are strongly NP-hard, an effort should be taken to solve the real size instances. According to that, this work addresses the genetic algorithm as the heuristic method to solve the problems. Genetic algorithm (GA) is an iterative population-based search method that leads to high search diversification, due to the use of a population of candidate solutions. Because of that, and because of the ability of the GA to allow worsening search steps to escape from local optimum, this method can well adapt to our problems, once it can find the best scheduling of trucks by the evolution of the population of candidate solutions through the promising regions. In this way, a good scheduling of trucks has higher survival probability and can generate offsprings with good genes in order to find the best configuration of trucks sequence.

We acknowledge that there are many other heuristics which could find best results or present better representation of the parallel machine problems with non-interference constraints than genetic algorithm. But at this work we have decided to carry only the genetic algorithm performance due to a time restriction.

Genetic algorithms belong to the class of evolutionary algorithms (EAs). These EAs provide a direct interaction with a population of candidate solutions. Evolutionary algorithms are a large and diverse class of algorithms based on the principles of natural evolution models of biological species. The principle of evolution is to find, through the steps of recombination, selection and mutation, the best species to generate optimal results.

An evolutionary algorithm basically starts with a set of candidate solutions (the initial population) and then performs a serie of three genetic operators: selection, mutation and recombination. In each EA's iteration, these operators are used to replace the current population by a new set of candidate solutions. The new population of each iteration is called *generation*.

EAs are a class of metaheuristic algorithms, which are most commonly used to solve NP-hard problems because they present a good performance (Gaiu, 2013). Within this class, the genetic algorithm is chosen to solve the analysed problem because it has been the most prominent type of EA for combinatorial problems (Bäck and Schütz, 1996) and (Hoos and Stützle, 2004). The GA has also been commonly applied to solve optimization problems in the field of industrial engineering (Gen and Cheng, 2000).

The GA works in the same way: it starts with a random initial population and then the three operators are performed: selection, crossover and mutation. Each candidate solution is called *chromosome*, and each population of chromosomes is called *population*. A simple GA presented by Mitchell (1998) works as follows:

Algorithm 1 General genetic algorithm

```
1: Randomly generate a population of $\mu$ chromosomes of $l$ bit
2: Calculate the fitness $f(\omega)$ of each chromosome $\omega$ of the population.
3: Repeat the following steps until $\mu$ offspring have been created:
     • Select a pair of parent chromosomes from the current population, the selection probability is an increasing function of fitness. Selection is done "with replacement", meaning that the same chromosome can be selected more than once.
     • With probability pc ("crossover probability"), cross over the pair at a randomly chosen point to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents.
     • Mutate the two offspring at each locus with probability pm ("mutation probability") and place the resulting chromosomes in the new population.
4: Replace the current population with the new population. If $\mu$ is odd, one new population member can be discarded at random.
5: Go to step 2.
```

The *selection* step is responsible to select the fittest chromosomes in the current population for the reproduction. At the reproduction step, the parents (a pair of chromosomes) are selected to *crossover*. The *crossover* randomly chooses a locus and exchanges the subsequences to create the offspring. The offspring generated can suffer *mutation* at some position according to some small probability.

Consider the example in which the chromosomes are jobs assignments to machines. Consider they are represented by a vector where the positions are the jobs and each position value corresponds to the job's machine assignment. Assume as well that the chromosomes are subjected to one-point crossover and mutation randomly reverses its bit values. For a problem with $M = 4$ machines and $J = 7$ jobs, we can have the chromosome 1312434, where job 1 is assigned to machine 1, job 2 to machine 3, and so on. If that chromosome is chosen to crossover the 4221134 chromosome at locus 3, for example, then the offspring generated are 1321134 and 4212434. If the last offspring suffers mutation at its second position, then it can turn into 4312434.

According to Gen and Cheng (2000), genetic algorithms work on two types of spaces alternatively: genotype space (coding space), and phenotype space (solution space). Genetic operators, such as crossover and mutation, work on genotype space, and evaluation and selection work on phenotype space. The GA provides a good balance between exploration and exploitation of the search space (Gen and Cheng, 2000). The genetic operators are responsible for exploring new regions of the space but can not guarantee the generation of improved offspring, because they perform a random search. On the other hand, the selection exploits the available information by directing the genetic search towards promising regions in the search space.

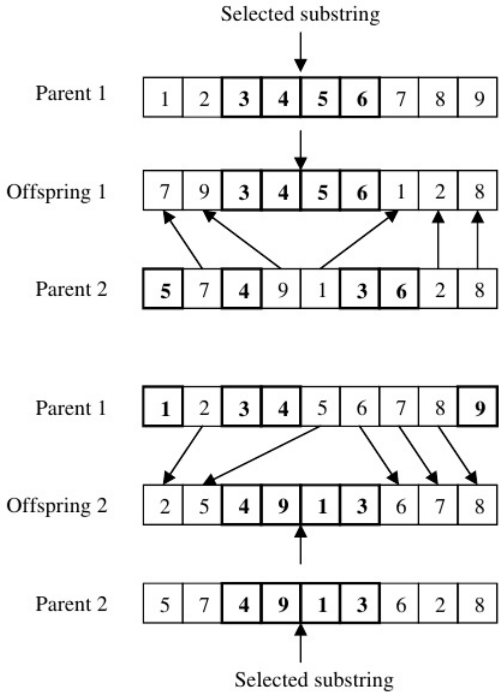
References about genetic algorithms and stochastic local search methods include the books of Michalewicz (1996), Mitchell (1998), Gen and Cheng (2000) and Hoos and Stützle (2004). Examples of genetic algorithms applied to solve parallel machine problems can be obtained on Glass et al. (1994), Min and Cheng (1999), Lee et al. (2008) and Chung and Choy (2012).

Glass et al. (1994) compare different local search methods for the $Rm||C_{max}$ problems, such as simulated annealing, tabu search and genetic algorithm. Their results show that genetic algorithms, implemented with genetic descent algorithm, are as good as simulated annealing and tabu search. Min and Cheng (1999) develop a genetic algorithm for the identical parallel machine problem with makespan minimization. Their GA configuration represents the assignment of jobs to machines and its solution's quality has advantage over the LPT rule and the simulated annealing method.

Lee et al. (2008) develop an efficient genetic algorithm in the case of quay crane scheduling with non-interference constraints and makespan minimization. They consider each chromosome as a sequence of jobs, which are scheduled on the machines in the same order. The assignment strategy of jobs to cranes considers the interference that it may cause. It also considers the earliest available crane and its position before processing the job. The authors adopt the order crossover, which ensures a valid sequence of jobs, and a mutation process that exchanges the place of two jobs in the sequence. The order crossover has the following major steps:

Step 1*:* randomly select a substring from one of the parents;
Step 2*:* copy the substring in the offspring at the same position that it is in the parent;
Step 3*:* copy the symbols of the second parent from left to right, excluding the symbols that are already in the substring.

These steps can be summarized in Figure 3.2. For other crossover methods please consult Gen and Cheng (2000).



Lee et al. (2008), p.p 132.

Figure 3.2: Representation of the order crossover method

Chung and Choy (2012) present a modified genetic algorithm for the $Pm|prec,r_j,p_j|w_1C_{max} + w_2\sum C_m$ problem, where C_m is the completion time of the quay crane m; and w_1 and w_2 are, respectively, the weight of the makespan and the weight of the sum of cranes completion time. Their chromosomes are represented by two segments: the first one is the jobs processing sequence, and the second one is the machine assignment. They are validated and corrected after each creation, in order to not violate the precedence and the non-interference constraints. Their results are slightly better than the ones found by ancient methods, but their proposed model is much faster.

3.2.1 Parameters setting

In order to find good solutions, the genetic algorithm must be set according to the particular problem. The design of an algorithm is an optimization problem itself, where parameters design strongly affect its performance. Defining appropriate parameters is one of the main challenges in evolutionary computing.

The parameters can be classified as *representation*, which defines the configuration of the candidate solution in a binary or some finite alphabet encoded chromosome; *evaluation function* of the chromosome, or *fitness*, which is related to the objective function. The fitness can also be a function of an associated penalty that can change according to the best feasible solution, the actual candidate solution and its violated constraints. Michalewicz (1995) reviews and tests six methods for handling constraints by genetic algorithms. According to the author, the process of evaluating an individual in a population may be quite complex, especially in the presence of feasible and infeasible solutions of a constrained problem.

Parent selection strategy is responsible for selecting the chromosomes to reproduction according to their fitness. *Population size* is the amount of chromosomes at each generation, which is usually empirically defined. According to Michalewicz (1996), populations too small may lead to quick convergence and too large may waste computational resources. Population size influences the population diversity and the selective pressure, because of that, the amount of offspring (λ) should be greater than the μ available parents to allow diversity (Gaiu, 2013). Eiben et al. (2007) suggest that putting effort into adapting the population size could be more effective than trying to adjust mutation and crossover operators.

However, according to Eiben et al. (2007), most studies on adaptive or self-adaptive EA's parameters concerns mutation and crossover operators. *Mutation operators* introduce modification into the population by inverting bits of the chromosome with probability *pm*. There has been significant efforts in finding static optimal values for mutation rates (Eiben et al., 1999). But Bäck (1992) shows that the mutation rate should not be constant and should decrease over time during the search. A comparison between a constant setting of mutation rate, a deterministic and a self-adaptation control schemes can be obtained in Bäck and Schütz (1996).

Giving a probability *pc* the *crossover operator* selects chromosomes that undergoes crossover following a certain mechanism. According to Eiben et al. (1999), crossover rate should not be too low and the value is rarely below 0.6. Crossover mechanism is the system responsible for reproduction, which includes, for example, the decision of the number of parents, the number and location of crossover points or the type of crossover. And finally, the *replacement operator* decides which chromosomes will belong to the population of the next generation in order to ensure a certain diversity of population.

These parameters are subdivided into qualitative and quantitative parameters. Qualitative ones are symbolic and define the main structure of an evolutionary algorithm, for example, crossover mechanism and parent selection structure. Quantitative parameters define a specific variant of this EA by setting parameter values, as for example mutation and crossover rates.

There are two approaches of setting parameter values: parameter tuning and parameter control. Parameter tuning is a common practice which defines the parameters values before the run of the algorithm and keeps them fixated while running. Their values are determined by iteratively generating and testing parameter vectors while seeking the one with high quality. This quality is called *utility* and is based on the performance and robustness of the EA, using the best parameter values. The insights about the EA with "optimal" parameters are obtained from the data collected while traversing the search space of parameter values.

In parameter tuning, the parameter values can be obtained by competitive testing, which aims to obtain an algorithm setup that meets some success criterion, as discussed by Eiben and Smit (2012). Or it can be obtained by scientific testing, which aims to gain insights into an EA through tuning algorithms and to provide superior parameter values. Tuning methods can facilitate well-funded experimental comparisons and algorithm analysis, as example of Meta-GA, REVAC, SPOT and F-Race. An extensive research on parameter tuning for evolutionary algorithms is given by Eiben and Smit (2011) and Eiben and Smit (2012). Gaiu (2013) evaluates the performance of tuning algorithms (REVAC, SPOT and F-Race) for the VRP (vehicle routing problem) solved by genetic algorithm and by multiple ant colony systems.

SPOT is the method selected to configurate the parameter vectors of the genetic algorithms addressed on this project, because of its high quality results. According to Eiben and Smit (2011) and Gaiu (2013), SPOT is one of the most efficient methods, as it is able to determine good parameter values while offering detailed information. Among other advantages, this method demands a low computational cost, requires the specification of few parameters and has an extensive documentation available. Bartz-Beielstein (2010) provides an explanation about SPOT, which is synthesized on Appendix A.

According to Eiben and Smit (2011), although the most promising developments on scientific testing happened after middle 2000s, parameter values are mostly conducted manually. On this context, the parameters values are not necessarily optimal, trying all the different combinations by hand is time-consuming and practically impossible. However, using tuning algorithms is highly rewarding, the efforts are moderate, the gains in performance can be very significant and it can obtain much information about parameter values and algorithm performance.

But, fixated parameters values are not in consonance with the dynamic and adaptative nature of genetic algorithms, the parameters should be modified during the run of the algorithm to lead to better performance. Eiben et al. (1999) and Eiben et al. (2007) demonstrate that different parameter values might be optimal in different stages of the evolutionary process.

Parameter control defines the change of parameters that influences the evolutionary process during the run. The definition of parameter control is even more difficult than defining static parameters, because they depend on the objective function and the used representation. Much research into this field has been done during the last decade (Eiben and Smit, 2011) and show that parameter control can speed up the convergence, improve robustness of evolutionary optimizations and avoiding suboptimal algorithm performance resulting from suboptimal parameter values set by the user.

The parameters values can be changed by a deterministic method if they are altered by a deterministic rule without considering the actual process; or by a feedback adaptation if some information can be gathered from the evolutionary process, such as population diversity measures, relative improvements, absolute solution quality, etc. These informations are used to determine the direction and the magnitude of the change in the strategy parameter, and to decide if the new parameters values propagate throughout the population.

Finally, the parameter values can be determined by self-adaptation, which is quite recent in the EA history (Eiben et al., 2007). In this principle the strategy parameters evolve with the evolutionary process. It exploits the indirect link between favorable strategy parameters and objective function values, such that the convergence velocity is optimized and the stochastic local optimization qualities of the algorithm are emphasized.

The behavior of evolutionary algorithms control parameters is complex. Examples on this field includes the works of Bäck (1992), Michalewicz (1995), Bäck and Schütz (1996), Lis and Lis (1996), Thierens (2002).

Bäck (1992) studies the convergence rates and the success probabilities for mutation-selection schemes, in which mutation is the only search operator in the GA. He addresses two types of selection: $(\mu, \lambda)-$selection, in which μ best individuals are selected from the λ offspring to become parents in the next generation; or $(\mu + \lambda)-$selection (elitist selection) in which μ best individuals are selected from the set of μ parents and λ offspring. The study shows that the elitist selection is advantageous for convergence velocity but disadvantageous when convergence reliability and self-adaptation of mutation rates are desired. It is observed that if the number of offspring increases, the expected progress increases. The selection schemes do not present difference for small mutation rates, but their progress decreases for growing rates. The author also gives a self-adaptation mechanims for controlling mutation in a bit-string GA.

Michalewicz (1995) reviews six methods for handling constraints by genetic algorithms for numerical optimization problems. These methods assume different ways to compute the penalties of the fitness function. The author tests the methods with five selected problems taking into account the type of the objective function, the number of variables and constraints, the types and the number of active constraints at the optimum, the ratio between the sizes of the feasible search space and the whole search space. Their strengths and weaknesses are then discussed.

Bäck and Schütz (1996) investigate the mutation rate in genetic algorithms using binary strings by comparing a constant mutation setting, a deterministic time-dependent and a self-adaptation mutation rate. The strengths of the proposed deterministic schedule and the self-adaptation method are demonstrated by a comparison of their performance on difficult combinatorial optimization problems. Both methods are shown to perform significantly better than the standard genetic algorithm, but the deterministic schedule yields the best average optimal objective function value and the lower number of runs to find the best optimum.

Lis and Lis (1996) propose a new dynamic method for parallel processing of genetic algorithms, which aims to control the mutation rate, crossover rate and population size during the run. The population is subdivided and evolves independently during some iterations (epoch). Their results are assemble by the algorithm and then re-divided. For each parameter a few possible probability values are defined in advance and three of them are chosen at the beginning of each epoch. Each subdivision executes the algorithm with a set of parameters that assumes only one of these three values. After each epoch the best results are evaluated: if it is given by the highest parameter, the probability of each subdivision is shifted up one level, other wise, it is shifted toward lower values.

Thierens (2002) evaluates two adaptive mutation rate control schemes in genetic algorithms: constant gain and declining adaptive rate control. The author shows their feasibility in comparison with a fixed mutation rate, a self-adaptive and a deterministic mutation rate. The schemes performance are evaluated into experiments with a counting ones problem and a zero/one multiple knapsack problem. The constant gain schemes performance is comparable to self-adaptive mutation, and has number of generations less sensitive to the initial mutation probability than fixed rate. The dynamic and declining adaptive schemes have comparable performance, they can match the optimal rate even when started from high initial *pm* values.

Chapter 4

Mathematical formulations

This chapter presents the mathematical models subject to each paradigm of non-interference constraints. Two unrelated parallel machine problems are proposed for the first paradigm, $MP1$ and $MP2$. And one is proposed for the second paradigm which is related to multiprocessors problem. The $MP1$ problem is formulated with precedence index variables and with time index variables, and is called $MP1_p$ and $MP1_t$ respectively. This relation is contextualized on Figure 4.1.

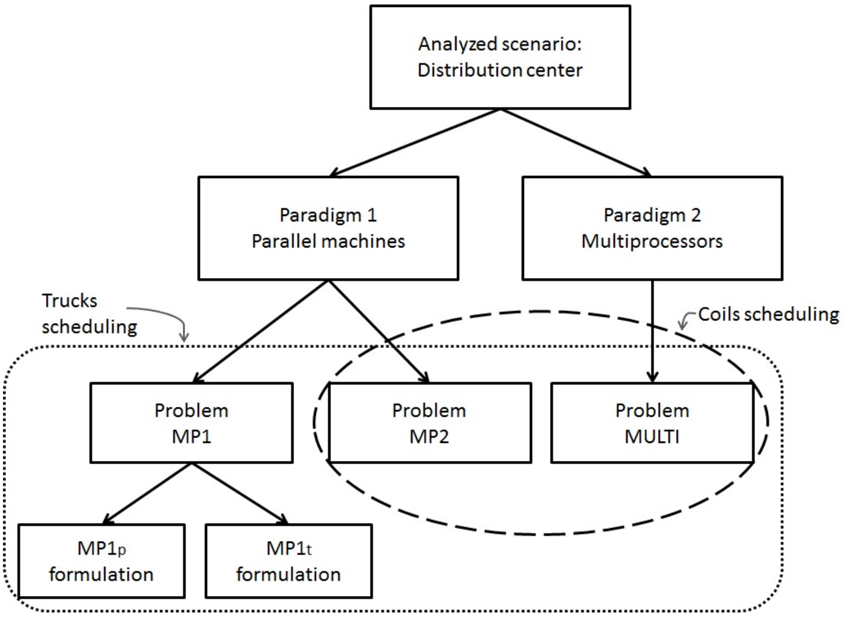


Figure 4.1: Machines configuration definition

All the presented models can well represent the problem. They are constructed with the same parameters, which are contextualized below. This chapter also presents the set of instances and the computational results of the formulations for different coil arrangements in the shed.

4.1 Notation

Before presenting the formulations, it is important to remember that the coils are the items, trucks are the jobs and cranes are the machines. Each coil is unique, i.e. there are not two similar coils in the shed. Without loss of generality, it is assumed that cranes and rows are indexed sequentially from left to right in increasing order. Each machine is considered to be always available to process jobs over a period of sequencing and they are not subject to disruptions, such as breakdowns, maintenance, among others. We assume that each job is available at time zero and must be completely processed without preemption. The cranes displacement are subject to interference because they move on the same trail. The processing time of each coil is composed by its retrieval time summed to the crane's displacement time between the truck and the row of that coil, and vice versa.

Given a set of items $\mathcal{B}$, that must be shipped by the set of machines $\mathcal{M} = \{1{,}2\}$. $Q_j \subseteq \mathcal{B}$, is the set of items that must be loaded by these machines on job $j \in \mathcal{J}$, where $\mathcal{J}$ is the set of available trucks. The items $i \in \mathcal{B}$ have machine dependent processing times $p_{ig} = p_0 + u_i^g$, where p_0 represents the retrieval time of the coils, which is the same for all of them. $g \in \{\mathcal{M} \bigcup \theta\}$ indicates the designated position of the truck processed by the machine $m \in \mathcal{M}$ or by both of them (θ).

Before explaining u_i^g, we assume that each item i has its row position in the shed given by l_i and that the trucks have to be positioned in l^g if it is processed by machine $g = m$ or by both of them $(g = \theta)$. Then u_i^g is given by the idled travel time from position l^g to position l_i summed to the loaded travel time from position l_i to l^g. The processing time of each job $j \in \mathcal{J}$ can be given by $p_{jg} = \sum_{i \in Q_j} p_{ig}$, and it is valid when considering no idle time between the processing of coils from the same truck. It is important to point out that $g = \theta$ is related to the multiprocessors paradigm, and it is not considered in the parallel machines configuration, where $g \in \{\mathcal{M} \bigcup \theta\}$ is substituted by $m \in \mathcal{M}$ to facilitate the undestanding.

To each job j is associated a $r_j^{min} = min_{i \in Q_j}(l_i)$ and a $r_j^{max} = max_{i \in Q_j}(l_i)$ which represents, respectively, the minimum and the maximum row of one of the coils from truck j. The parameter

w_j is the weight, or priority factor, of a job j and Δ is the safety operational distance of the cranes. G is a very large integer given by the time horizon H, where $H = \sum_{i \in \mathcal{B}} max_{m \in \mathcal{M}}(p_{im})$ and $\bar{G}$ is a large integer given by the amount of rows in the shed plus one. The problem is to find completion times C_j for all jobs $j \in \mathcal{J}$ with respect to the constraints such that the total weighted completion time is minimized.

4.2 Parallel machine paradigm

Parallel machine paradigm considers that each truck is processed by only one of the cranes, independently of the amount of coils. This paradigm is represented by two problems: $MP1$ and $MP2$ configurations, which are subject to non-interference constraints.

4.2.1 MP1 configuration

The unrelated parallel machines configuration $MP1$ disregards the removal sequence of each coil assigned to a truck, and bounds an area in the storage shed for the loading. I.e., when processing a job j, the processing of other trucks $f \in \mathcal{J}$ is prevented in an area bounded by the rows of the demanded coils of job j. This problem is formulated with precedence index variables ($MP1_p$ configuration) and with time index variables ($MP1_t$ configuration) in order to evaluate the best type of mathematical formulation.

Formulation with precedence index variables

In the first formulation, we consider binary linear ordering variables, such as in the works of Rocha et al. (2008) and Lee et al. (2008). The variables s_{fj} are equal to 1 when job f precedes job j, and equal to 0 otherwise; and $y_{jm} = 1$ if crane m processes job j, and equal to 0 otherwise. The start time of the truck j is computed by $t_j \geq 0$. The model is written as follow.

$$Min \sum_{j \in \mathcal{J}} w_j * (t_j + \sum_{m \in \mathcal{M}} p_{jm} y_{jm}) \tag{4.1}$$

$$\sum_{m \in \mathcal{M}} y_{jm} = 1, \ \forall j \in \mathcal{J} \tag{4.2}$$

$$t_f + p_{fm} \leq t_j + G(1 - y_{fm}) + G(1 - y_{jm}) + G(1 - s_{fj}),$$

$$\forall f \in \mathcal{J}, \ \forall j \in \mathcal{J} \mid f \neq j, \ \forall m \in \mathcal{M} \tag{4.3}$$

$$t_j + p_{jm} \leq t_f + G(1 - y_{fm}) + G(1 - y_{jm}) + G s_{fj},$$

$$\forall f \in \mathcal{J}, \ \forall j \in \mathcal{J} \mid f \neq j, \ \forall m \in \mathcal{M} \tag{4.4}$$

$$t_f + \sum_{m \in \mathcal{M}} p_{fm} y_{fm} - t_j + G s_{fj} > 0, \ \forall f \in \mathcal{J}, \ \forall j \in \mathcal{J} \tag{4.5}$$

$$t_f + \sum_{m \in \mathcal{M}} p_{fm} y_{fm} - t_j - G(1 - s_{fj}) \leq 0, \ \forall f \in \mathcal{J}, \ \forall j \in \mathcal{J} \tag{4.6}$$

$$\bar{G}(s_{fj} + s_{jf}) \geq y_{jm} r_j^{max} - y_{fm+1}(r_f^{min} - \Delta), \ \forall f \in \mathcal{J}, \ \forall j \in \mathcal{J} \mid f \neq j, \ \forall m \in \mathcal{M} \tag{4.7}$$

$$y_{jm} \in \{0{,}1\}, \ \forall j \in \mathcal{J}, \ \forall m \in \mathcal{M} \tag{4.8}$$

$$s_{fj} \in \{0{,}1\}, \ \forall f \in \mathcal{J}, \ \forall j \in \mathcal{J} \tag{4.9}$$

$$t_j \geq 0, \ \forall j \in \mathcal{J} \tag{4.10}$$

The objective function is given by (4.1) and should minimize the total weighted completion time of the jobs. The set of constraints (4.2) states that each truck must be processed by exactly one machine. Constraints (4.3) and (4.4) compute the start time of each truck j. The sets of constraints (4.5) and (4.6) define the values for the precedence variable s_{fj}. (4.5) forces $s_{fj} = 1$ when f is concluded before j starts, independently of the machine where they are being processed. But (4.5) does not force $s_{fj} = 0$ when the jobs f and j are simultaneously processed, or when j precedes f. The set of constraints (4.6) is supplementary to (4.5) and forces $s_{fj} = 0$ when precedence does not occur. The set of constraints (4.7) prevents interference between the cranes. If $(s_{fj} + s_{jf}) = 0$ then the jobs f and j are processed simultaneously by the machines. The sets of constraints (4.8) to (4.10) define the domain of the variables.

Formulation with time index variables

For the second formulation, we consider time index variables, in which the planning horizon H is discretized into several periods, where period t starts at time $t = 0$ and ends at time $t = H$. We introduce a binary time index variable, y_{jt}^{m} which is equal to 1 if truck j begins its processing at time t by crane m, and it is equal to 0 otherwise. Then the same problem can be modeled as:

$$Min \sum_{j \in \mathcal{J}} w_j C_j \tag{4.11}$$

$$\sum_{m \in \mathcal{M}} \sum_{t=0}^{H-p_0} y_{jt}^{m} = 1, \ \forall j \in \mathcal{J} \tag{4.12}$$

$$y_{jt}^{m} + \sum_{z=t}^{min(t+p_j^m-1, H-p_f^m)} y_{fz}^{m} \leq 1, \ \forall m \in \mathcal{M}, \ \forall t \in \mathcal{H}, \ \forall j \in \mathcal{J}, \ \forall f \in \mathcal{J} \mid j \neq f \tag{4.13}$$

$$\sum_{m \in \mathcal{M}} \sum_{t=0}^{H-p_0} (t + p_j^m) y_{jt}^{m} - C_j = 0, \ \forall j \in \mathcal{J} \tag{4.14}$$

$$y_{jt}^{m} + \sum_{z=max(0,t-p_v^n+1)}^{min(t+p_j^m-1, H-p_f^n)} y_{fz}^{n} \leq 1, \ \forall j \in \mathcal{J},$$

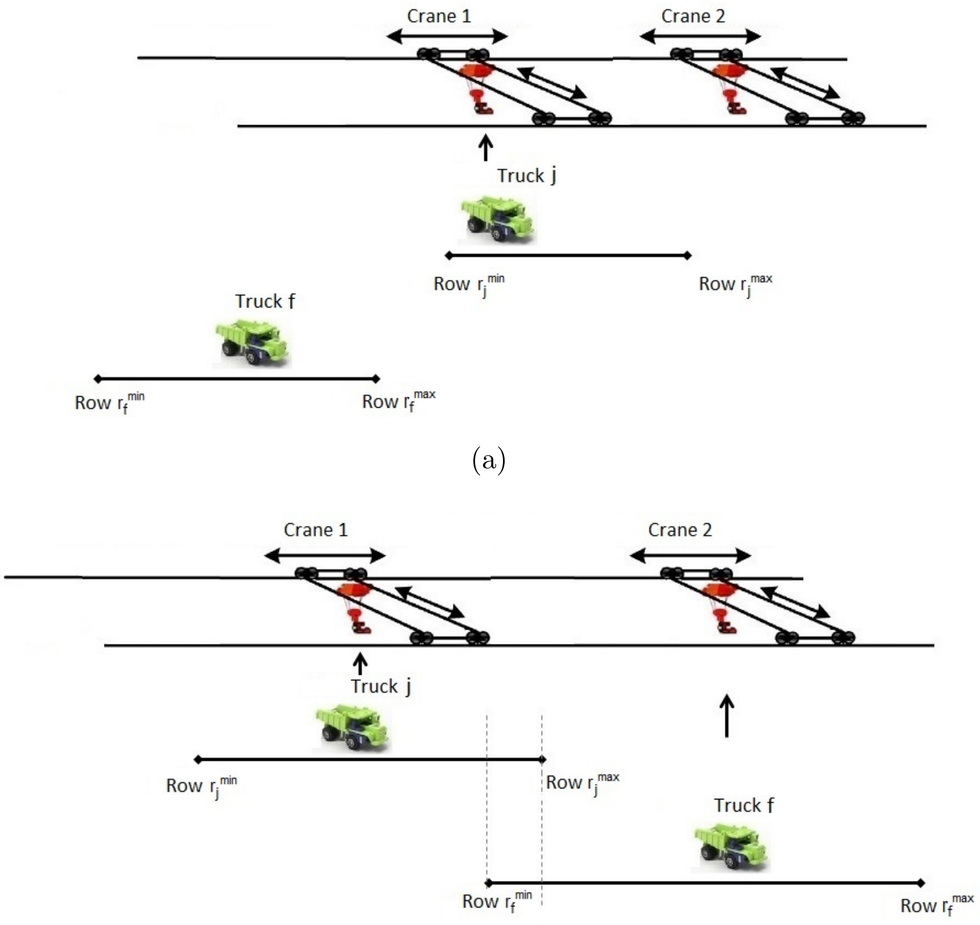
$$\forall f \in \mathcal{J}, \ \forall t \in \{0, \ldots, H - p_j^m\}, \ \forall m \in \mathcal{M}, \ \forall n \in \mathcal{M} \mid n > m, \ r_j^{max} \geq (r_f^{min} - \Delta) \tag{4.15}$$

$$y_{jt}^{m} \in \{0,1\}, \ \forall j \in \mathcal{J}, \ \forall t \in \mathcal{H}, \ \forall m \in \mathcal{M} \tag{4.16}$$

$$C_j \geq 0, \ \forall j \in \mathcal{J} \tag{4.17}$$

The objective function is given by (4.11) and should minimize the total weighted completion time of the jobs. The set of constraints (4.12) ensures that each truck is processed by only one machine at a period of time. The constraints (4.13) certify that at each time only one truck is performed by each machine and (4.14) compute the completion time of each truck j.

The set of constraints (4.15) prevents interference between cranes, Figures 4.2 (a) and (b). Consider that truck f has its minimum row leftmost the maximum row of truck j ($r_j^{max} > r_f^{min} + \Delta$, where Δ is the safety distance). When truck j is processed on machine 1, then no truck f can be processed by machine 2 during the operational time of j by crane 1. Figure 4.2 (a) shows how the set of constraints (4.15) prevents the crossing of cranes; and Figure 4.2 (b) shows how such set of constraints prevents the intercession of the trucks' areas. The sets of constraints (4.16)



(a)

(b)

Figure 4.2: Explanatory example to the constraints (4.15). r_j^{min} and r_j^{max} indicate, respectively, minimum and maximum row of a coil from truck j.

and (4.17) define the domain of the model variables.

Comparison between the formulations

The evaluation of the two formulations are executed with different set of parameters. The instances were obtained from the load allocation model of da Silva Neto (2013) who considers three levels of availability of trucks, which can assume different values for each quantity of products. The availability are classified as *very restrictive* (vr), *somewhat restrictive* (sr), and *not restrictive* (nr) availability of trucks. As considered by da Silva Neto (2013), we also consider that the coils are randomly arranged in the shed.

For each group of instances, 10 replications were generated with different seeds, totaling 150 artificial instances. Furthermore, it is considered a safety distance of one row, and w_j is generated from the uniform distribution $[0, 1]$. The average retrieval time of a coil is 4, which was obtained from the data collected at a distribution center of a steelmaker company. The travel time between n consecutive rows is presented in Table 4.1.

Table 4.1: Ranges of displacement times

range of displacement	one way displacement time	complete displacement time
between 1 and 19 rows	0.5 minutes	1 minute
between 20 and 39 rows	1.5 minutes	3 minutes
between 40 and 58 rows	2.5 minutes	5 minutes
between 59 and 78 rows	3.5 minutes	7 minutes

Further explanation about the parameters generation are given on Section 4.4. Both mathematical models were implemented and solved using the CPLEX 12.5 optimization software in the default configuration, through the AMPL language. The computer used in the tests is an Intel (R) Xeon (R) CPU X5690 @ 3.47GHz with 24 processors, 132 GB of RAM in Ubuntu Linux operating system. The runs were terminated after one hour of CPU time.

Small instances were used to evaluate the performance of both models, which must find the same optimal solution value. To compare the different formulations we look at the number of test cases unsolved within one hour of CPU time, the average computational time of solved instances, the number of explored B&B nodes and the gap of optimality of feasible non-optimal solutions.

Table 4.2 is a compilation of the results of the ordering formulation and the time index formulation. The first column indicates the group of instances, the $\sum S^*$ column is the amount of instances solved optimally; S^* represents the average weighted completion time found in the optimal cases; S^f represents the average weighted completion time found in the feasible non-optimal cases after one hour of CPU time.

S^{*f} column presents the average solution values found in the optimal cases of the time index formulation for the instances which give feasible non-optimal solutions for the precedence formulation. For example, for the group of instances *nr_30* of the $MP1_t$, S^* represents the average of the solutions found by the time index formulation of the same 9 instances considered at the S^* column of $MP1_p$, and S^{*f} represents the optimal $MP1_t$ solution for the same instance considered in S^f of $MP1_p$.

CT (s) represents the average computation time; *Node* is the average amount of B&B nodes explored by CPLEX and *GAP* indicates the distance between the current best integer solution and the best bound found within the time limit.

Table 4.2: Average results of the time index formulation ($MP1_t$) and the precedence index formulation ($MP1_p$). $\sum S^*$: amount of instances solved optimally; S^*: average weighted completion time found in the optimal cases; S^f: average solution values found in the feasible non-optimal cases after 1h of CPU time; S^{*f}: average solution values found in the optimal cases of the $MP1_t$ for the instances which give feasible non-optimal solutions for the $MP1_p$. CT: average CPU time; Node: average B&B nodes explored by CPLEX; GAP: average optimality gap.

Inst.	$MP1_p$						$MP1_t$						
	$\sum S^*$	S^*	S^f	CT (s)	Node	GAP	$\sum S^*$	S^*	S^{*f}	S^f	CT (s)	Node	GAP
nr_10	10	42.5		0	0	0.0%	10	42.5		0.0	1	0	0.0%
sr_10	10	47.6		0	0	0.0%	10	47.6		0.0	2	0	0.0%
vr_10	10	49.5		1	0	0.0%	10	49.5		0.0	2	0	0.0%
nr_15	10	71.3		1	743	0.0%	10	71.3		0.0	2	39	0.0%
sr_15	10	85.5		2	5602	0.0%	10	85.5		0.0	3	39	0.0%
vr_15	10	107.3		7	36707	0.0%	10	107.3		0.0	3	109	0.0%
nr_20	10	129.8		1	4066	0.0%	10	129.8		0.0	12	0	0.0%
sr_20	10	114.6		90	384750	0.0%	10	114.6		0.0	15	1	0.0%
vr_20	10	139.6		122	415815	0.0%	10	139.6		0.0	15	0	0.0%
nr_25	9	194.3	254.5	49	1101710	22.4%	10	194.3	254.5	0.0	40	10	0.0%
sr_25	10	161.7		161	773150	0.0%	10	161.7		0.0	33	0	0.0%
vr_25	8	186.2	258.2	32	1835365	21.5%	10	186.2	257.4	0.0	41	4	0.0%
nr_30	9	252.5	345.4	749	3016753	27.6%	10	252.5	345.4	0.0	58	27	0.0%
sr_30	4	250.3	344.8	491	4398141	31.7%	10	250.3	344.1	0.0	174	789	0.0%
vr_30	5	253.3	321.7	428	3376496	32.7%	10	253.3	321.2	0.0	109	860	0.0%

Although the traditional parallel machine problem can be usually better formulated by precedence index variables, the results showed that the parallel machine problem with non-interference constraints does not present a good performance with this formulation. As expected, in all optimal cases, the solutions of the formulations are exactly the same. The time indexed formulation demands fewer CPU time and can solve all the small instances within 1 hour of CPU time, while the ordering indexed approach can not solve some instances with more than 25 coils within the time limit. Furthermore, the time index formulation explores fewer nodes of Branch & Bound than the precedence index formulation. Because of that, the $MP1_t$ is the formulation considered in the $MP1$ problem.

As can be seen, in the set of instances *nr_25* there is one instance that is not called optimal and has a 22.4% GAP. The solution found for this instance is the same in both formulations, but the CPLEX was not able to prove the optimality for this solution in the first formulation. The same happens for some other instances where the optimality was not proved after one hour of run.

4.2.2 MP2 configuration

The uniform parallel machines configuration $MP2$ considers that each machine shall completely process a truck. The problem must schedule the processing of trucks and each one of their coils. A new decision variable is defined as x_{it}^{em}, which is equal to 1 if coil i is assigned to position e and starts in t by crane m, otherwise it equals zero. Position data e is the available processing order for each truck.

$$\textbf{Min} \sum_{j\in\mathcal{J}} w_j C_j \tag{4.18}$$

$$\sum_{j\in\mathcal{J}} \sum_{e=1}^{q_j} \sum_{m\in\mathcal{M}} \sum_{t=0}^{H-p_i^m} x_{it}^{em} = 1, \ \forall i \in \mathcal{B} \tag{4.19}$$

$$\sum_{m\in\mathcal{M}} \sum_{t=0}^{H-p_i^m} \sum_{i\in Q_j} x_{it}^{em} = 1, \ \forall j \in \mathcal{J}, \ \forall e \in \{1,\ldots,q_j\} \tag{4.20}$$

$$\sum_{e\in\mathcal{B}} x_{vt}^{em} + \sum_{j\in\mathcal{J}} \sum_{e=1}^{q_j} \sum_{z=t}^{min(t+p_v^m-1,H-p_i^m)} x_{iz}^{em} \leq 1,$$

$$\forall m \in \mathcal{M}, \ \forall t \in \{0,\ldots,H-p_v^m\}, \ \forall v \in \mathcal{B}, \ \forall i \in \mathcal{B} \mid v \neq i \tag{4.21}$$

$$\sum_{m\in\mathcal{M}} \sum_{t=0}^{H-p_0} y_{jt}^m = 1, \ \forall j \in \mathcal{J} \tag{4.22}$$

$$y_{jt}^m - \sum_{i\in Q_j} x_{it}^{1m} = 0, \ \forall m \in \mathcal{M}, \ \forall j \in \mathcal{J}, \ \forall t \in \{0,\ldots,H-p_0\} \tag{4.23}$$

$$x_{it}^{em} - \sum_{v\in Q_j|i\neq v} x_{vmin(t+p_i^m,H-p_v^m)}^{e+1m} \leq 0, \ \forall m \in \mathcal{M}, \ \forall j \in \mathcal{J},$$

$$\forall i \in Q_j, \ \forall e \in \{1,\ldots,q_j-1\}, \ \forall t \in \{0,\ldots,H-p_i^m\} \mid q_j \geq 2 \tag{4.24}$$

$$\sum_{e\in\mathcal{B}} x_{it}^{em} + \sum_{j\in\mathcal{J}} \sum_{e=1}^{q_j} \sum_{z=max(0,t-p_v^n+1)}^{min(t+p_i^m-1,H-p_v^n)} x_{vz}^{en} \leq 1,$$

$$\forall t \in \{0,\ldots,H-p_i^m\}, \ \forall v \in \mathcal{B}, \ \forall i \in \mathcal{B}, \ \forall m \in \mathcal{M}, \ \forall n \in \mathcal{M} \mid n > m, \ l_i \geq l_v - \Delta \tag{4.25}$$

$$\sum_{m\in\mathcal{M}} \sum_{t\in 0,\ldots,H-p_0} (t + \sum i \in Q_j p_i^m) y_{jt}^m - C_j = 0, \ \forall j \in \mathcal{J} \tag{4.26}$$

$$x_{it}^{em} \in \{0,1\}, \ \forall i \in \mathcal{B}, \ \forall j \in \mathcal{J}, \ \forall e \in \{1,\ldots,q_j\}, \ \forall t \in \mathcal{H}, \ \forall m \in \mathcal{M} \tag{4.27}$$

$$y_{jt}^m \in \{0,1\}, \ \forall j \in \mathcal{J}, \ \forall t \in \mathcal{H}, \ \forall m \in \mathcal{M} \tag{4.28}$$

$$C_j \geq 0, \ \forall j \in \mathcal{J} \tag{4.29}$$

The objective function is given by (4.18) and aims to minimize the total weighted completion time of the jobs. The set of constraints (4.19) ensures that each item is processed only once; (4.20) requires that each coil of a truck is made at only one processing position. This can be exemplified on Figure 4.3, in which a truck needs to be loaded with three coils, thereby, one of its coils (called coil a) might be processed firstly, secondly or thirdly.

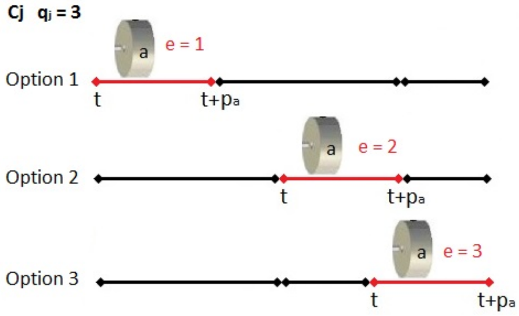


Figure 4.3: Explanatory example to the constraints (4.20)

The set of constraints (4.21) certifies that each machine processes at most one item in each period. The constraints (4.22) ensure that each job is processed only once. Constraints (4.23) require item $i \in Q_j$ of position $e = 1$ to begin exactly when its job starts to be processed. This relationship can be verified on Figure 4.4, where coil $i \in Q_j$ of position $e = 1$ begins exactly when truck j starts.

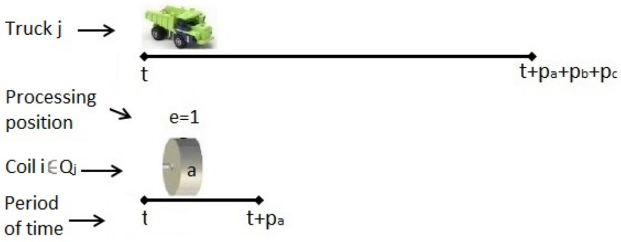


Figure 4.4: Explanatory example to the constraints (4.23)

The set of constraints (4.24) ensures that: the coils of the same truck j are performed without idle time or interruption to process a coil $v \ni Q_j$ between the processing of coils $i \in Q_j$. Figure 4.5 shows an example of a truck with 3 coils processed in sequence, without preemption.

Constraints (4.26) compute the completion time of each truck j. The set of constraints (4.25) prevents interference on the cranes displacement, illustrated on Figure 4.6 (a). If coil i is processed on machine 1 , then any coil v , which is in a leftmost row ($l_i > l_v$) , can be processed by machine 2 while coil i is performed by crane 1. Remember that rows and cranes are indexed on increasing order from left to right in the shed. Similarly, Figure 4.6 (b), if coil v

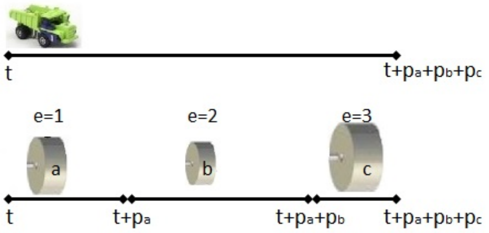
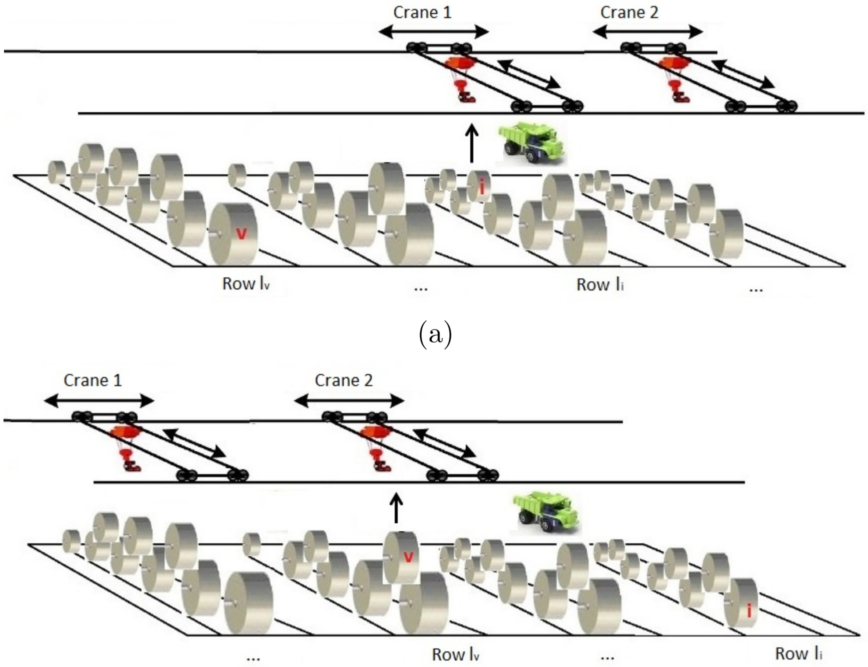


Figure 4.5: Explanatory example to the constraints (4.24)

is processed by the crane 2 , then no coil i, such that $l_i > l_v$, can be processed by machine 1. The sets of constraints (4.27) to (4.29) define the domain of the model variables.



(a)

(b)

Xie et al. (2014), p.p. 2875, adapted.

Figure 4.6: Explanatory example to the constraints (4.25)

4.3 Multiprocessors paradigm

4.3.1 MULTI configuration

In the multiprocessor configuration ($MULTI$) each truck is simultaneously processed by two identical machines. At this problem, we assume parameter $\bar{q}_j$ which is the half amount of coils demanded by truck j. This parameter aims to divide the amount of coils from truck j between the two cranes. S_k is the group of machines contained into the set of machines k, such that

$k \in \mathcal{K} = \{1,\ldots,K\}$, where K is the amount of groups of machines.

The $MULTI$ model is an extension of the $MP2$ model, where the decision variables x_{it}^{em} are defined in the same way, i.e. they are equal to 1 if coil i, that is processed at the position e, starts processing in t by the crane m, or zero otherwise. The y variables are changed by y_{jt}^{k}, which are equal to 1 if truck j begins processing in t by the set of machines k, or 0 otherwise. $\bar{C}_i \geq 0$ represents the completion time of coil i.

$$\textbf{Min} \sum_{j \in \mathcal{J}} w_j C_j \tag{4.30}$$

$$\sum_{j \in \mathcal{J}} \sum_{e=1}^{q_j} \sum_{m \in \mathcal{M}} \sum_{t=0}^{H} x_{it}^{em} = 1, \ \forall i \in \mathcal{B} \tag{4.31}$$

$$\sum_{i \in Q_j} \sum_{m \in \mathcal{M}} \sum_{t=0}^{H-p_i^*} x_{it}^{em} = 1, \ \forall j \in \mathcal{J}, \ \forall e \in \{1,\ldots,q_j\} \tag{4.32}$$

$$\sum_{j \in \mathcal{J}} \sum_{e=1}^{q_j} x_{vt}^{em} + \sum_{j \in \mathcal{J}} \sum_{e=1}^{q_j} \sum_{z=t}^{min(t+p_v^*-1,H-p_i^*)} x_{iz}^{em} \leq 1,$$

$$\forall m \in \mathcal{M}, \ \forall v \in \mathcal{B}, \ \forall i \in \mathcal{B}, \ \forall t \in \{0,\ldots,H-p_v^*\} \mid v \neq i \tag{4.33}$$

$$\sum_{k=1}^{K} \sum_{t=0}^{H-p_0} y_{jt}^{k} = 1, \ \forall j \in \mathcal{J} \tag{4.34}$$

$$\sum_{i \in Q_j} \sum_{k=1}^{K-1} \sum_{t=0}^{H-p_i^*} y_{jt}^{k} = 1, \ \forall j \in \mathcal{J} \mid q_j = 1 \tag{4.35}$$

$$\sum_{t=0}^{H-p_0} y_{jt}^{K} = 1, \ \forall j \in \mathcal{J} \mid q_j \geq 2 \tag{4.36}$$

$$y_{jt}^{k} - \sum_{m \in S_k} \sum_{i \in Q_j} x_{it}^{1m} \leq 0, \ \forall k \in \{1,\ldots,K\}, \forall j \in \mathcal{J}, \ \forall t \in \{0,\ldots,H-p_0\} \tag{4.37}$$

$$x_{jt}^{1m} - \sum_{v \in Q_j} \sum_{z=t}^{min(t+p_i^*,H-p_v^*)} x_{vz}^{\bar{q}_j+1n} \leq 0, \ \forall j \in \mathcal{J},$$

$$\forall t \in \{0,\ldots,H-p_0\}, \ \forall i \in Q_j, \ \forall m \in \mathcal{M}, \ \forall n \in \mathcal{M} \mid m \neq n, \ q_j \geq 2 \tag{4.38}$$

$$x_{it}^{em} - \sum_{v \in Q_j | i \neq v} x_{v min(t+p_i^*,H-p_v^*)}^{e+1m} \leq 0, \ \forall j \in \mathcal{J}, \ \forall e \in \{1,\ldots,\bar{q}_j - 1\},$$

$$\forall i \in Q_j, \ \forall t \in \{0,\ldots,H-p_i^*\}, \ \forall m \in \mathcal{M} \mid q_j \geq 3 \tag{4.39}$$

$$x_{it}^{em} - \sum_{v \in Q_j | i \neq v} x_{vmin(t+p_i^*, H-p_v^*)}^{e+1m} \leq 0, \; \forall j \in \mathcal{J}, \; \forall e \in \{\bar{q}_j + 1, \ldots, q_j - 1\},$$

$$\forall i \in Q_j, \; \forall t \in \{0, \ldots, H - p_i^*\}, \; \forall m \in \mathcal{M} \mid q_j \geq 3 \tag{4.40}$$

$$\sum_{e \in \mathcal{B}} x_{it}^{em} + \sum_{e \in \mathcal{B}} \sum_{z=max(0,t-p_v^*+1)}^{min(t+p_i^*-1, H-p_v^*)} x_{vz}^{en} \leq 1, \; \forall v \in \mathcal{B},$$

$$\forall i \in \mathcal{B}, \; \forall t \in \{0, \ldots, H - p_i^*\}, \; \forall m \in \mathcal{M}, \; \forall n \in \mathcal{M} \mid n > m, \; (l_i \geq l_v - \Delta) \tag{4.41}$$

$$\bar{C}_i - x_{it}^{em}(t + p_i^*) \geq 0, \; \forall j \in \mathcal{J},$$

$$\forall e \in \{1, \ldots, q_j\}, \; \forall i \in Q_j, \; \forall t \in \{0, \ldots, H - p_i^3\}, \; \forall m \in \mathcal{M} \tag{4.42}$$

$$\bar{C}_i - C_j \leq 0, \; \forall j \in \mathcal{J}, \; \forall i \in Q_j \tag{4.43}$$

$$x_{it}^{em} \in \{0,1\}, \; \forall i \in \mathcal{B}, \; \forall j \in \mathcal{J}, \; \forall e \in \{1, \ldots, q_j\}, \; \forall t \in \mathcal{H}, \; \forall m \in \mathcal{M} \tag{4.44}$$

$$y_{jt}^k \in \{0,1\}, \; \forall j \in \mathcal{J}, \; \forall t \in \mathcal{H}, \; \forall k \in \{1, \ldots, K\} \tag{4.45}$$

$$C_j \geq 0, \; \forall j \in \mathcal{J} \tag{4.46}$$

$$\bar{C}_i \geq 0, \; \forall i \in \mathcal{B} \tag{4.47}$$

The objective function is given by (4.30) and aims to minimize the weighted completion time of jobs. Constraints (4.31) ensure that each coil is processed only once and (4.32) require that only one coil is processed at each position of the truck's processing. The set of constraints (4.33) certifies that each machine processes at most one coil at each period. Constraints (4.34) ensure that each truck is processed only once. The set of constraints (4.35) requires that only one machine processes trucks with only one coil, but it does not specify which is the machine. The set of constraints (4.36) ensures that trucks with more than one coil are processed by more than one machine.

The set of constraints (4.37) forces coil at position 1 to begin when its truck starts processing. To explain the constraints (4.38) consider a truck with $q_j \geq 2$, then half of its coils must be processed on crane m (coils between positions 1 and $\bar{q}_j$) and the other half on machine n (coils between positions $\bar{q}_j + 1$ and q_j). Thus (4.38) states that the coil at the first position on machine n (position $\bar{q}_j + 1$) shall start while the coil on position 1 is performed, Figure 4.7 exemplifies this relation.

Considering the assumption described above, the constraints (4.39) and (4.40) ensure that coils of the same truck are processed without preemption. The non-interference constraints (4.41) are

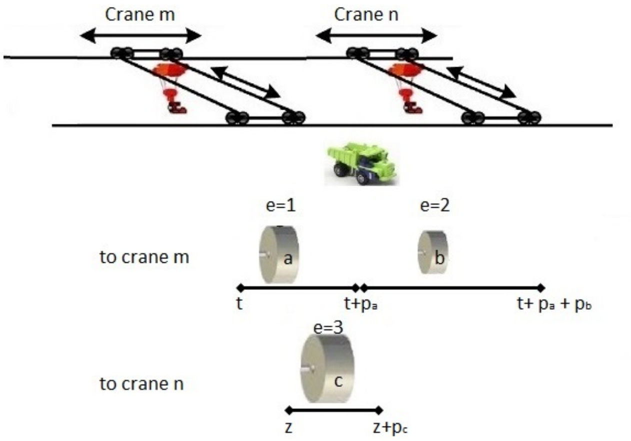


Figure 4.7: Explanatory example to the constraints (4.38)

identical to the non-interference constraints of the $MP2$ model and can be explained exactly on the same way. The constraints (4.42) and (4.43) compute, respectively, the completion time of each coil i and the completion time of each truck j. Constraints (4.44) to (4.47) set the model's domain variables.

4.4 Computational experiments

Several sets of instances were defined to analyze the performance of the paradigms. At most one input is changed at each group of instance.

4.4.1 Instances definition

The instances were obtained from the load allocation model of da Silva Neto (2013), who developed instances considering real data provided by a steelwork company from Minas Gerais, Brazil, between November 2011 and March 2012. To evaluate the mathematical models and the heuristic performance are considered 15 groups of artificial instances: there are five amounts of coils, $\{10, 20, 50, 100, 200\}$, and three levels of trucks availability. These levels are defined for each amount of coils and are identified as *very restrictive* (vr), *somewhat restrictive* (sr) and *not restrictive* (nr). The very restrictive level represents the low availability of trucks and the "nr" level represents the wide availability. These definitions are used in the generation of our instances. Their uniform distribution can be verified on Table 4.3, in which the minimum and maximum values are specified in $U(min, max)$.

Table 4.3: Availability of trucks by amount of products and restrictive level

	Level of restriction according to trucks availability:		
amount of products	very restrictive	somewhat restrictive	not restrictive
10	U(1,3)	U(2,6)	200
20	U(1,5)	U(6,10)	200
50	U(3,8)	U(10,15)	200
100	U(7,12)	U(15,20)	200
200	U(15,20)	U(30,40)	200

da Silva Neto (2013), p.p 44, adapted.

Although the model developed by da Silva Neto (2013) considers that coils are randomly arranged in the shed (scenario R), this study also addresses two other cases: the storage of coils into rows belonging to their client (scenario C), and storing into rows belonging to a client group (scenario CG). The amount of rows demanded by the set of products of each client is defined in function to its share in the distribution center.

In scenario C, the coils storage considers the client who they belong to. It is defined the amount of rows demanded to store coils of the same client. These coils are then grouping in the same row(s). In scenario GC, the coils storage considers a group of customers. The storage shed of the distribution center is divided into 10 parts and the clients' demands are accumulated. The groups are built considering the aggregate demand of customers: in group 1 there are clients in which the sum of their coils' demand does not exceed 10% of the available rows; group 2 is formed with clients to whom the sum of their aggregate demands does not exceed 20%; and so on. For each configuration 10 replications are generated with different seeds, totaling 450 artificial instances.

The average retrieval time of each coil is 3.4 minutes. This value was obtained from the data collected at a distribution center of a steelmaker company, where a total of 40 trucks and 103 steel coils were monitored. The average retrieval time disregards the information about the level of the coil in the row, because the considered scenario does not have this data. Thus, the average simplification tends to nullify the errors associated with this measure. Intending to be conservative, the average retrieval time is rounded upwards to 4 minutes / coil.

The setup time between the consecutive loading of two coils is not considered, but we consider the displacement time spent by the cranes to move between the truck and the row of the coil, and between the row of the coil and its truck. The displacement time added to the retrieval time, results in the processing time of each coil which varies from coil to coil and the designated position of the truck.

To calculate the displacement time, the trucks are assumed to be in a designated position: rows 23 and 70, respectively, for the cranes 1 and 2 of the problems from the parallel machines paradigm; and on row 42 for the multiprocessors paradigm. This study considers 95 rows in the shed. It is assumed the distance of 2 meters between two consecutive rows. The cranes speed, 38.5 meters / minute, is resulting from 50% of the average of maximum speeds provided by three distributors of such equipments (Igus, Eurocrane and Schneider-Electric). In this way, the travel time depends on the range of displaced rows, presented on Table 4.4.

Table 4.4: Travel time according to the ranges of displaced rows

range of displacement	one way displacement time	complete displacement time
between 1 and 19 rows	0.5 minutes	1 minute
between 20 and 39 rows	1.5 minutes	3 minutes
between 40 and 58 rows	2.5 minutes	5 minutes
between 59 and 78 rows	3.5 minutes	7 minutes

The displacement time of the first range, for example, is the result of the average travel times through one row, two rows, and so on, until 19 rows of distance between the coil and the designated place of the truck. The same applies to the other ranges of rows. The crane always starts on the designated position of the truck (e.g. if crane 1, then the truck designated position is row 23, as defined previously) and moves toward the row of the demanded coil and returns to its designated position. Because of that, the displacement time is considered twice, as shown on third column of Table 4.4.

Coils of the same truck are made without preemption and trucks are loaded without idle time between this process. The safety distance of the cranes displacement is one row. The time horizon is given by the sum of the maximum processing time of the coils. The priority weight of each truck is randomly generated by a uniform distribution between zero and one. The parameters can be verified on Table 4.5.

Table 4.5: Parameters values for the paradigms' evaluation

Parameter	Value
amount of coils	$\{10, 20, 50, 100, 200\}$
retrieval time of each coil (p_0)	4
time horizon (H)	$\sum_{i \in \mathcal{B}} max_{m \in \mathcal{M}} p_i^m$
weight (w_j)	U(0,1)
safety distance (Δ)	1 row

To analyze the difference between the formulations, we evaluate the solutions generated by the models after 3600 seconds of CPU time. The proposed mathematical models were implemented and solved using the CPLEX 12.5 optimization software in the default configuration, it was used the AMPL language. The computer used in the tests is an Intel (R) Xeon (R) CPU X5690 @ 3.47GHz with 24 processors, 132 GB of RAM in Ubuntu Linux operating system.

4.4.2 Results of the mathematical models

This section aims to present the results generated by the three problems. It is expected that $MP1$ model find optimal values greater than or equal to the $MP2$ optimal values, because the bounding area defined by the rows of the coils of each truck makes the cranes have more idle time. To compare between the different formulations, it is observed the number and the results of tested cases with optimal solutions, feasible non-optimal solutions or without solution after 1 hour of CPU time. As well as the computational time spent by the optimal cases, the average number of explored nodes and the average optimality gap given by the solver within 1 hour of CPU time.

The Tables 4.6, 4.7 and 4.8 are a compilation of the results of $MP1$, $MP2$ and $MULTI$ problems for three scenarios. Scenario R represents the random arrangement of coils in the storage shed. On scenario C the coils are arranged by customers' row(s) and on scenario GC by a customer grouping arrangement. The performance of each model is evaluated in order to identify which one best fits each kind of storage scenario.

The first column of the tables represents the instance, which is codified by the availability of trucks: nr, sr and vr, respectively level not restrictive, somewhat restrictive and very restrictive; and by the amount of steel coils: $\{10, 20, 50, 100, 200\}$. The $\sum S^*$ column is the amount of instances solved optimally; $\sum nb$ is the amount of instances without basis after one hour of CPU time. S^* column presents the average weighted completion time found in the optimal cases; S^f represents the average weighted completion time found in the feasible non-optimal cases after one hour of CPU time. *CT (s)* represents the average computation time of the optimal solutions; if this field contains "-" the time limit of 3600 seconds is reached. *Node* is the average amount of B&B nodes explored by CPLEX for the optimal and non-optimal cases; and GAP indicates the average distance between the current best integer solution and the best bound found within the time limit for the feasible non-optimal solutions.

In the $MP1$ partition, the column S^{*2} represents the average of optimal solution values given by $MP1$ model for the same instances optimally solved by $MP2$, or by $MULTI$ when appropriate. S^{*f} represents the average of optimal results given by $MP1$ for the instances which are feasible non-optimally solved by $MP2$ or $MULTI$ model. The column S^{*nb} represents the average of optimal solutions given by $MP1$ for instances that could not be solved by $MP2$ problem, or by $MULTI$ when appropriate. E.g., for the group of instances *sr_20* at the random scenario, Table 4.6, S^{*2} represents the average of solutions values found by $MP1$ for the same 4 instances optimally solved by $MP2$ and considered in its S^* column. S^{*f} represents the optimal solution given gy $MP1$ for the instance that is non-optimally solved by $MP2$; and S^{*nb} represents the average of optimal solutions of $MP1$ for the 5 instances without basis when solved by $MP2$.

The analysis of the results shows that model $MP1$ is able to find optimal solutions faster than $MP2$ and $MULTI$. $MP2$ finds slightly better optimal solutions than $MP1$, but $MP1$ has better computational performance [1]. $MP1$ can find optimal solutions for instances of up to 100 coils within the time limit. The $MP1$ model optimally solve all the tested instances up to 20 coils, and all instances of 50 coils of the scenarios C and GC. The $MP1$ model can optimally solve 67% of the instances of 50 coils of the R scenario, and 35% of the instances of 100 coils of C and GC scenarios. $MP2$ can solve optimally only 69% of the instances up to 20 coils and can not find solutions for larger instances within the time limit. $MULTI$ can solve optimally only 57% of the instances of 10 coils and can not find solutions for 93% of the instances of 20 coils. While the $MP1$ can not find solutions for 43% of the instances of 200 coils.

In general, the non-optimal feasible solutions found by $MP2$ for instances of 20 coils are worse than those found by $MP1$. However, the solutions found by $MP2$ on GC scenario for instances up to 20 coils are better than those found optimally by $MP1$. Both models present better optimal solutions for the scenario GC and worst solutions for the scenario R. The $MULTI$ model finds better solutions than $MP2$ for scenarios R and GC, but worst ones for scenario C. This is because on this scenario the coils of the same truck are in the same row(s), thus, in order to avoid interference, the machines must have more idle times.

The results show that $MP2$ and $MULTI$ models can optimally solve very small problems, while $MP1$ can solve small and medium size ones, however with a dramatic CPU time increased for

[1] The reader must remember that the MP1, MP2 and MULTI, are different problems. Thus different optimal solutions are expected.

larger instances. When the amount of coils increases from 10 units to 50, the CPU time of the $MP1$ increases by 690 times, $MP2$ is unable to find feasible solutions for 22% of the instances with 20 coils, and $MULTI$ can not find feasible solutions for 93% of these instances within the time limit. $MP2$ and $MULTI$ are unable to find feasible solutions for instances greater than 20 coils. This shows the importance of developing techniques for improving the computational performance when solving medium or larger size instances.

Throughout the analysis and validation process, we have decided to no continue to expand the MULTI model, as it presents a higher complexity and the worst performance. On its place, it may be considered the single machine paradigm, since it is possible to admit two cranes as a single processor. In this way, both cranes could process each truck, regardless the number of coils in each one. In order to solve real size instances, we decided to work only with the parallel machine paradigm by developing two genetic algorithms.

Table 4.6: Random Scenario (Scenario R) - Average results of the $MP1$, $MP2$ and $MULTI$ problems.

	$MP1$										$MP2$						
Inst.	$\sum S^*$	$\sum nb$	S^*	S^f	S^{*2}	S^{*f}	S^{*nb}	CT	Node	GAP	$\sum S^*$	$\sum nb$	S^*	S^f	CT	Node	GAP
nr_10	10	0	42.5		42.5			1	0	0%	10	0	38.7		21	99	0%
sr_10	10	0	47.6		47.6			1	0	0%	10	0	44.5		16	50	0%
vr_10	10	0	49.5		49.5			1	0	0%	10	0	47.1		10	86	0%
nr_20	10	0	129.8			138.4	124.1	11	0	0%	0	6		259.8	-	4	54.2%
sr_20	10	0	114.6		140.2	103.2	96.3	14	1	0%	4	5	136.2	98.0	1030	258	2.4%
vr_20	10	0	139.6		146.2	150.4	58.4	14	0	0%	4	1	139.2	142.5	1236	635	5.3%
nr_50	6	0	522.3	706.8			522.3	1416.5	5245.2	3%		10			-		
sr_50	7	0	608.4	765.7			608.4	1610.6	1286.0	9%		10			-		
vr_50	7	0	583.6	779.2			583.6	1076.4	5709.9	7%		10			-		
nr_100	0	0		5084.2				-	0.0	56%		10			-		
sr_100	0	0		4624.0				-	0.0	55%		10			-		
vr_100	0	0		3937.4				-	0.0	44%		10			-		
nr_200	0	6		79087.8				-	0.0	100%		10			-		
sr_200	0	10						-				10			-		
vr_200	0	10						-				10			-		

	$MP1$										$MULTI$						
Inst.	$\sum S^*$	$\sum nb$	S^*	S^f	S^{*2}	S^{*f}	S^{*nb}	CT	Node	GAP	$\sum S^*$	$\sum nb$	S^*	S^f	CT	Node	GAP
nr_10	10	0	42.5		38.6	46.3		1	0	0%	5	0	32.7	38.0	1724	28593	52.7%
sr_10	10	0	47.6		36.3	64.7		1	0	0%	6	0	33.9	58.6	1576	54124	58.7%
vr_10	10	0	49.5		46.1	54.6		1	0	0%	6	0	43.6	53.6	891	98011	44.7%
nr_20	10	0	129.8				129.8	11	0	0%	0	10			-		
sr_20	10	0	114.6			145.6	111.1	14	1	0%	0	9		348.2	-	80	98.4%
vr_20	10	0	139.6			176.4	135.5	14	0	0%	0	9		292.0	-	1031	97.4%
nr_50	6	0	522.3	706.8			522.3	1416.5	5245.2	3%		10			-		
sr_50	7	0	608.4	765.7			608.4	1610.6	1286.0	9%		10			-		
vr_50	7	0	583.6	779.2			583.6	1076.4	5709.9	7%		10			-		
nr_100	0	0		5084.2				-	0.0	56%		10			-		
sr_100	0	0		4624.0				-	0.0	55%		10			-		
vr_100	0	0		3937.4				-	0.0	44%		10			-		
nr_200	0	6		79087.8				-	0.0	100%		10			-		
sr_200	0	10						-				10			-		
vr_200	0	10						-				10			-		

Table 4.7: Scenario of customers rows (Scenario C) - Average results of the $MP1$, $MP2$ and $MULTI$ problems

	$MP1$										$MP2$						
Inst.	$\sum S^*$	$\sum nb$	S^*	S^f	S^{*2}	S^{*f}	S^{*nb}	CT	Node	GAP	$\sum S^*$	$\sum nb$	S^*	S^f	CT	Node	GAP
nr_10	10	0	40.6		40.6			1	0		10	0	37.6		13	8	
sr_10	10	0	42.1		42.1			1	0		10	0	41.8		12	21	
vr_10	10	0	46.2		46.2			1	0		10	0	46.1		11	16	
nr_20	10	0	105.6		102.9	98.6	129.3	7	0		2	2	101.1	131.4	1342	183	22.2%
sr_20	10	0	103.7		120.4	90.2	82.3	9	0		5	2	119.2	89.2	749	177	1.5%
vr_20	10	0	127.7		125.3	133.2		10	0		7	0	124.3	150.6	781	411	13.8%
nr_50	10	0	479.4				479.4	196	144			10			-		
sr_50	10	0	521.2				521.2	267	80			10			-		
vr_50	10	0	549.8				549.8	249	592			10			-		
nr_100	3	0	1859.0	1975.6			1859.0	2201	5422	2.0%		10			-		
sr_100	3	0	1911.2	1813.8			1911.2	2015	3506	2.2%		10			-		
vr_100	4	0	1818.7	1926.2			1818.7	1618	2975	1.9%		10			-		
nr_200	0	0		79399.3				-	0.0	100.0%		10			-		
sr_200	0	5		82230.5				-	0.0	100.0%		10			-		
vr_200	0	8		84475.5				-	0.0	100.0%		10			-		
	$MP1$										$MULTI$						
Inst.	$\sum S^*$	$\sum nb$	S^*	S^f	S^{*2}	S^{*f}	S^{*nb}	CT	Node	GAP	$\sum S^*$	$\sum nb$	S^*	S^f	CT	Node	GAP
nr_10	10	0	40.6		35.3	46.0		1	0		5	0	32.7	40.7	1417	27456	59.4%
sr_10	10	0	42.1		38.3	47.8		1	0		6	0	40.0	57.5	1380	54577	45.5%
vr_10	10	0	46.2		42.6	51.6		1	0		6	0	49.2	69.7	543	70218	54.8%
nr_20	10	0	105.6				105.6	7	0		0	10			-		
sr_20	10	0	103.7			139.9	99.7	9	0		0	9		290.2	-	2	97.9%
vr_20	10	0	127.7			146.7	125.5	10	0		0	9		386.8	-	2	97.9%
nr_50	10	0	479.4				479.4	196	144			10			-		
sr_50	10	0	521.2				521.2	267	80			10			-		
vr_50	10	0	549.8				549.8	249	592			10			-		
nr_100	3	0	1859.0	1975.6			1859.0	2201	5422	2.0%		10			-		
sr_100	3	0	1911.2	1813.8			1911.2	2015	3506	2.2%		10			-		
vr_100	4	0	1818.7	1926.2			1818.7	1618	2975	1.9%		10			-		
nr_200	0	0		79399.3				-	0.0	100.0%		10			-		
sr_200	0	5		82230.5				-	0.0	100.0%		10			-		
vr_200	0	8		84475.5				-	0.0	100.0%		10			-		

Table 4.8: Scenario of grouping customers rows (Scenario GC) - Average results of the $MP1$, $MP2$ and $MULTI$ problems

	$MP1$										$MP2$						
Inst.	$\sum S^*$	$\sum nb$	S^*	S^f	S^{*2}	S^{*f}	S^{*nb}	CT	Node	GAP	$\sum S^*$	$\sum nb$	S^*	S^f	CT	Node	GAP
nr_10	10	0	39.7		39.7			1	0		10	0	37.3		15	8	
sr_10	10	0	41.6		41.6			1	0		10	0	41.3		13	28	
vr_10	10	0	46.1		46.1			1	0		10	0	45.9		10	52	
nr_20	10	0	104.4		111.2	102.4	108.2	8	0		1	2	111.2	116.7	1307	384	19.6%
sr_20	10	0	101.8		109.9	83.6	82.0	8	0		7	1	108.8	106.7	1049	441	21.9%
vr_20	10	0	127.6		128.9	128.0	119.9	11	0		5	1	126.0	124.1	1178	201	3.1%
nr_50	10	0	482.0				482.0	345	2131			10			-		
sr_50	10	0	522.1				522.1	311	648			10			-		
vr_50	10	0	552.2				552.2	261	1603			10			-		
nr_100	3	0	1822.1	1994.3			1822.1	2613	3140	1.7%		10			-		
sr_100	4	0	1946.7	1725.1			1946.7	2416	2774	2.1%		10			-		
vr_100	4	0	1819.2	1923.2			1819.2	2000	4328	2.2%		10			-		
nr_200	0	0		85679.2				-	0.0	100%		10			-		
sr_200	0	0		84471.8				-	0.0	100%		10			-		
vr_200	0	0		85628.8				-	0.0	100%		10			-		
	$MP1$										$MULTI$						
Inst.	$\sum S^*$	$\sum nb$	S^*	S^f	S^{*2}	S^{*f}	S^{*nb}	CT	Node	GAP	$\sum S^*$	$\sum nb$	S^*	S^f	CT	Node	GAP
nr_10	10	0	39.7		43.1	34.7		1	0		4	0	28.8	43.3	1333	33558	54.4%
sr_10	10	0	41.6		44.8	36.8		1	0		6	0	36.1	62.3	1730	71490	59.4%
vr_10	10	0	46.1		47.2	42.2		1	0		7	0	53.6	62.9	634	44331	50.0%
nr_20	10	0	104.4				103.1	8	0		0	10			-		
sr_20	10	0	101.8			71.4	102.0	8	0		0	9		345.3	-	3	98.1%
vr_20	10	0	127.6			115.4	132.4	11	0		0	9		413.7	-	2	98.0%
nr_50	10	0	482.0				482.0	345	2131			10			-		
sr_50	10	0	522.1				522.1	311	648			10			-		
vr_50	10	0	552.2				552.2	261	1603			10			-		
nr_100	3	0	1822.1	1994.3			1822.1	2613	3140	1.7%		10			-		
sr_100	4	0	1946.7	1725.1			1946.7	2416	2774	2.1%		10			-		
vr_100	4	0	1819.2	1923.2			1819.2	2000	4328	2.2%		10			-		
nr_200	0	0		85679.2				-	0.0	100%		10			-		
sr_200	0	0		84471.8				-	0.0	100%		10			-		
vr_200	0	0		85628.8				-	0.0	100%		10			-		

Chapter 5

Genetic algorithms

Two genetic algorithm are implemented to solve the probems addressed by the parallel machine paradigm. It is called $GA1$ when applied to solve the $MP1$ problem and $GA2$ to solve the $MP2$ problem. This chapter aims to present the two genetic algorithms structures, explain and define their strategy parameters and present the results obtained with the tuned parameters.

5.1 Parameters definition

The genetic algorithms presented in this chapter are constructed based in the approach of Lee et al. (2008). The algorithm developed by the authors can solve the crane scheduling problem to fill the holds of vessels effectively and efficiently, with a gap lower than 2.7% from the lower bound. Thus, our genetic algorithm is based on their work, arbitrarily fixing some algorithmic structure and parameters. However, we are aware that there exist a multitude of genetic algorithms settings and, recognizing that other algorithm structures may lead to better results.

Each candidate solution is called *chromosome*, each group of chromosomes is called *population* and each iteration of the algorithm is called *generation*. Following the approach of Lee et al. (2008), we represent each chromosome as a permutation encoding which is a string of a sequence of numbers. In this way, each candidate solution of the $GA1$ represents a possible solution for the sequence of J trucks; the chromosome of the $GA2$ represents the possible solution for the trucks sequencing and their coils sequencing (defined on Section 5.3).

Fitness is a quality function of the chromosomes, which indicates the best species acording to their results. Fitness is a simple transformation of the objective function, which is calculated by a simple *assignment heuristic*. It assigns jobs to machines following the sequence given by the chromosome and considering the non-interference constraints. In this dissertation, we work only with feasible solutions. This function is not penalised because the constraints are guaranteed to be satisfied during the scheduling of the given candidate solution. Since the goal is to find the problem's minimum solution, GA increases the survival chances of individuals with higher fitness. The fitness function is given by:

$$f(\omega) = \frac{1}{\sum_{j \in \mathcal{J}} w_j C_j}$$

where C_j is the completion time of job $j \in \mathcal{J}$ and w_j is the priority factor of job j.

To create the next generation, a pair of chromosomes (*parents*) are selected to create the offspring, i.e. they are selected to *crossover*. The offspring generated can suffer *mutation* at some position according to some probability *pm*. Also following the approach of Lee et al. (2008), the parents are selected to crossover by roulette wheel selection. In this method, the selection probability is proportional to the fitness value of each chromosome and the selection of parents considers the replacement of the already selected ones. An order crossover is applied to generate two offsprings. It is done by randomly choosing a substring (subsequence of the sequence of trucks) where the beginning and the tail of the substring are chosen with equal uniform probability. The mutation mechanism consists of randomly choosing two points, or trucks, and change their positions into the sequence.

5.1.1 Parameters tuning

In order to obtain good solutions in feasible computation time, some parameters are tuned for the proposed GAs by the SPOT method (Appendice A). To assure a good balance between exploration, exploitation and diversity of the search space, the following components of the genetic algorithm are determined by a parameter tuning algorithm.

Mutation and crossover operators are tested because they are responsible for exploring new regions by a random search; and population selection is tested because it exploits the available information by directing the genetic search towards promising evolution, (Gen and Cheng, 2000). Eiben et al. (1999) and Eiben and Smit (2011) consider these as the main components of evolutionary algorithms. Population size is also evaluated because of its influence in the population diversity and selective pressure. And the number of generations is evaluated due to its influence in the algorithm's computational time.

The GA inputs of the SPOT tuning algorithm are the lower and upper bound of the parameter values which will be defined in the following. The population size is assumed as a function of the instance size ($\mu = dB/2$) where B is the amount of coils that must be delivered and d is an integer that varies between 1 and 10. In the same way, the number of generations is assumed as a function of B, $N_{max} = gB$ where g is an integer that varies between 1 and 50. These ranges are assumed because the reviwed works consider very different population levels and amount of generations, for example, Lee et al. (2008) consider population size of 300 and 1000 generations for instances with up to 35 holds and 4 quay cranes; and Chung and Choy (2012) considers population size of 50 and 30000 generations for instances with up to 25 tasks and 3 quay cranes.

Replacement operator is related to the selection of chromosomes to survive in the next generation. In this work, the SPOT algorithm evaluates the GA's performance between two deterministic procedures that select the best chromosomes from parents and offspring: $(\mu + \lambda)-$selection and $(\mu, \lambda)-$selection. According to Gen and Cheng (2000), these methods prohibit selection of duplicate chromosomes from a population. As defined by Bäck (1992), $(\mu + \lambda)-$selection is an elitist selection and $(\mu, \lambda)-$selection addresses the complete replacement of the parents by the best individuals of the offspring.

The lower bound of pc in the tunning algorithm is assumed equal to 0.6, rate indicated by Eiben et al. (1999) as the lower limit for the crossover probability, and the upper bound is equal to 1, as it is not specified an upper limit. To tune mutation probability it is considered $pm \in [0.001, 0.01]$. This is the range of small values considered by most of the genetic algorithms works (Bäck and Schütz, 1996).

5.2 Genetic algorithm to solve $MP1$

The unrelated parallel machine problem $MP1$ disregards the removal sequence of each coil from each truck and bounds an area in the storage shed delimited by the processing of truck $j \in \mathcal{J}$ (from r_j^{min} to r_j^{max}, as defined in Chapter 4). The genetic algorithm developed to solve $MP1$ is called $GA1$ and has five parameters tuned to improve its performance. The tuned parameters indicated by the analisys of the results of SPOT (Appendix A), are given in the last column of Table 5.1.

Table 5.1: Parameters tuned the GA1

	Evaluated region		
Parameter	Lower bound	Upper bound	Indicated value
Population size (μ)	$B/2$	$10B$	$8B$
Replacement operator (ro)	$0 = (\mu + \lambda)-$selection	$1 = (\mu, \lambda)-$selection	1
Crossover probability (pc)	0.6	1	0.9705
Mutation probability (pm)	0.001	0.01	0.0037
Amount of generations (N_{max})	$1B$	$50B$	$31B$

where B is the size of the instance, given by the amount of coils to be delivered

The best parameter values defined by SPOT tuning algorithm are: population size (μ) equals $8B/2$ and amount of generations (N_{max}) equals $31B$, where B represents the size of the instance. Replacement operator (ro) given by $(\mu, \lambda)-$selection, crossover probability (pc) and mutation probability (pm) equal 97.05% and 0.37%, respectively. These parameters help to characterize the genetic algorithm summarized on Algorithm 2.

Algorithm 2 Genetic algorithm for the MP1 configuration

```
1: Randomly generate a population of $\mu$ chromosomes of $J$ trucks.
2: Calculate the fitness $f(\omega)$ of each chromosome $\omega$ of the population.
3: Create $\lambda$ offspring.
     • Select a pair of parents.
     • Generate two offspring with order crossover probability $pc$.
     • With mutation probability $pm$, two randomly chosen points of the sequence change
       their position.
4: Replace the population.
5: Return to step 2.
```

Fitness is a function of the weighted completion time value of the candidate solution. The weighted completion time of each chromosome is computed by assigning each truck to the cranes following the sequence of them given by the chromosome. This heuristic is defined in the following section.

5.2.1 Assignement heuristic

This is a simple heuristic, developed to compute the solution value of the assignment of jobs to machines, following the sequence given by each chromosome. Consider ct_m and ct_n as the current completion time of machine m and n, both $\in \mathcal{M}$, and $ct_m = ct_n = 0$ at time zero. Consider the set $\bar{\mathcal{J}} \subseteq \mathcal{J}$ of unassigned jobs. The crane schedule can be built using the procedure given on Figure 5.1.

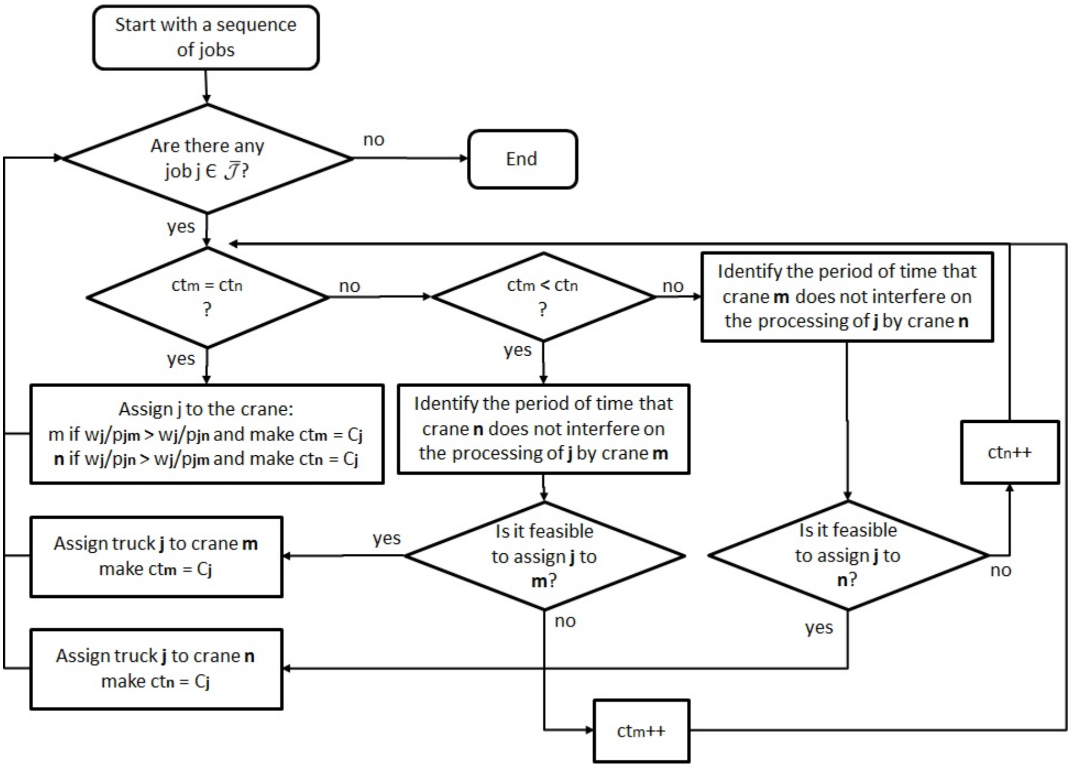

Start with a sequence of jobs
Are there any job j ∈ $\bar{\mathcal{J}}$?
no
End
yes
ctm = ctn ?
no
ctm < ctn ?
no
Identify the period of time that crane **m** does not interfere on the processing of **j** by crane **n**
yes
yes
Assign j to the crane: m if wj/pjm > wj/pjn and make ctm = Cj **n** if wj/pjn > wj/pjm and make ctn = Cj
Identify the period of time that crane **n** does not interfere on the processing of **j** by crane **m**
ctn++
Assign truck **j** to crane **m** make ctm = Cj
yes
Is it feasible to assign **j** to **m**?
Is it feasible to assign **j** to **n**?
no
no
yes
Assign truck **j** to crane **n** make ctn = Cj
ctm++

Figure 5.1: Flowchart of the proposed sequencing heuristic for the GA1

The jobs are assigned to the first available machine following the same order given by the trucks sequence of a chromosome. This scheduling process considers the non-interference constraints. E.g. if machine m is the first one available, and if the assignment of the next job j to machine m leads to interference on the working space of machine n, then, the algorithm considers that machine m should have an idle time instead of processing j in order to avoid interference. If both machines are available, then the job j is assigned to machine m if the factor $w_j/p_{jm} > w_j/p_{jn}$, or to n otherwise, where w_j is the weight of job j and p_{jm} is the processing time of truck j on machine m.

5.3 Genetic algorithm to solve $MP2$

As defined previously, the genetic algorithm to solve the $MP2$ problem is called $GA2$. It considers the scheduling of trucks and their coils, in which the interference between cranes is a function of the coil position in the shed (l_i). The candidate solution of the $GA2$ is represented as a permutation encoding of the possible solution for the trucks sequencing and their coils sequencing, as presented on Figure 5.2.

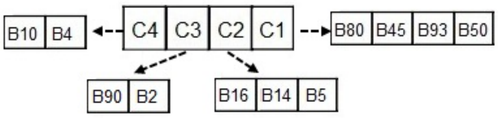


Figure 5.2: Example of GA2 chromosome representation

where Cf is the ID of truck f and Bl is the ID of the coil that is on row l. E.g. $\{C4,\ C3,\ C2,\ C1\}$ is a possible scheduling of trucks and $\{B16,\ B14,\ B5\}$ is a possible retrieval sequence of the coils from truck $C2$.

To build the $GA2$, the selection of parents, replacement, mutation and crossover operators are the same as defined previously. However, it is assumed that the sequence of demanded coils of each truck is also subject to order crossover with probability pc and to mutation with probability pm. We consider that the coils sequence subject to crossover and mutation are those from the trucks that do not belong to the substring delimited by the order crossover of the trucks' sequence, Figure 5.3 exemplify this procedure.

With probability pm, we assume that two randomly chosen coils of a truck change their position into the coils sequence. And with the same probability, two randomly chosen trucks change their sequencing position. The tuned parameters indicated by the analisys of the results of SPOT (Appendix A), are given on Table 5.2.

Table 5.2: Parameters tuned the GA2

	Evaluated region		
Parameter	Lower bound	Upper bound	Indicated value
Population size (μ)	$B/2$	$10B$	10
Replacement operator (ro)	$0 = (\mu + \lambda)-$selection	$1 = (\mu, \lambda)-$selection	0
Crossover probability (pc)	0.6	1	0.8973
Mutation probability (pm)	0.001	0.01	0.0064
Amount of generations (N_{max})	$1B$	$50B$	$46B$

where B is the size of the instance, given by the amount of coils to be delivered

The best parameter values defined by SPOT tuning algorithm are: population size (μ) equals

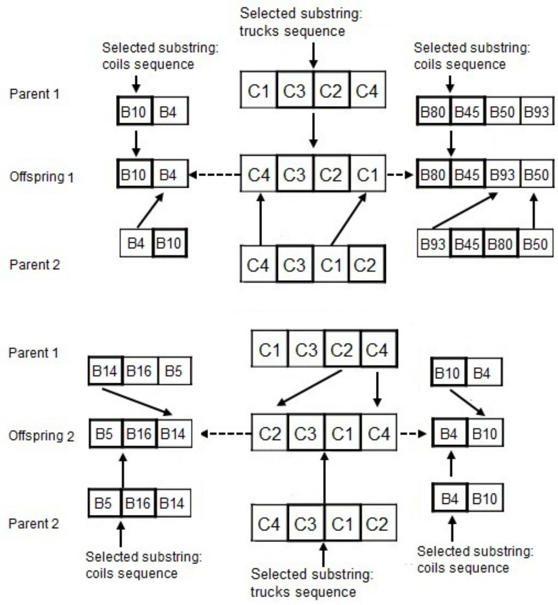


Figure 5.3: Example of trucks crossover considering coils crossover

$10B/2$ and number of generations (N_{max}) equals $46B$, where B represents the size of the instance. Replacement operator (ro) given by $(\mu+\lambda)-$selection, crossover probability (pc) and mutation probability (pm) equal 89.73% and 0.64%, respectively. These parameters help characterize $GA2$ summarized on Algorithm 3.

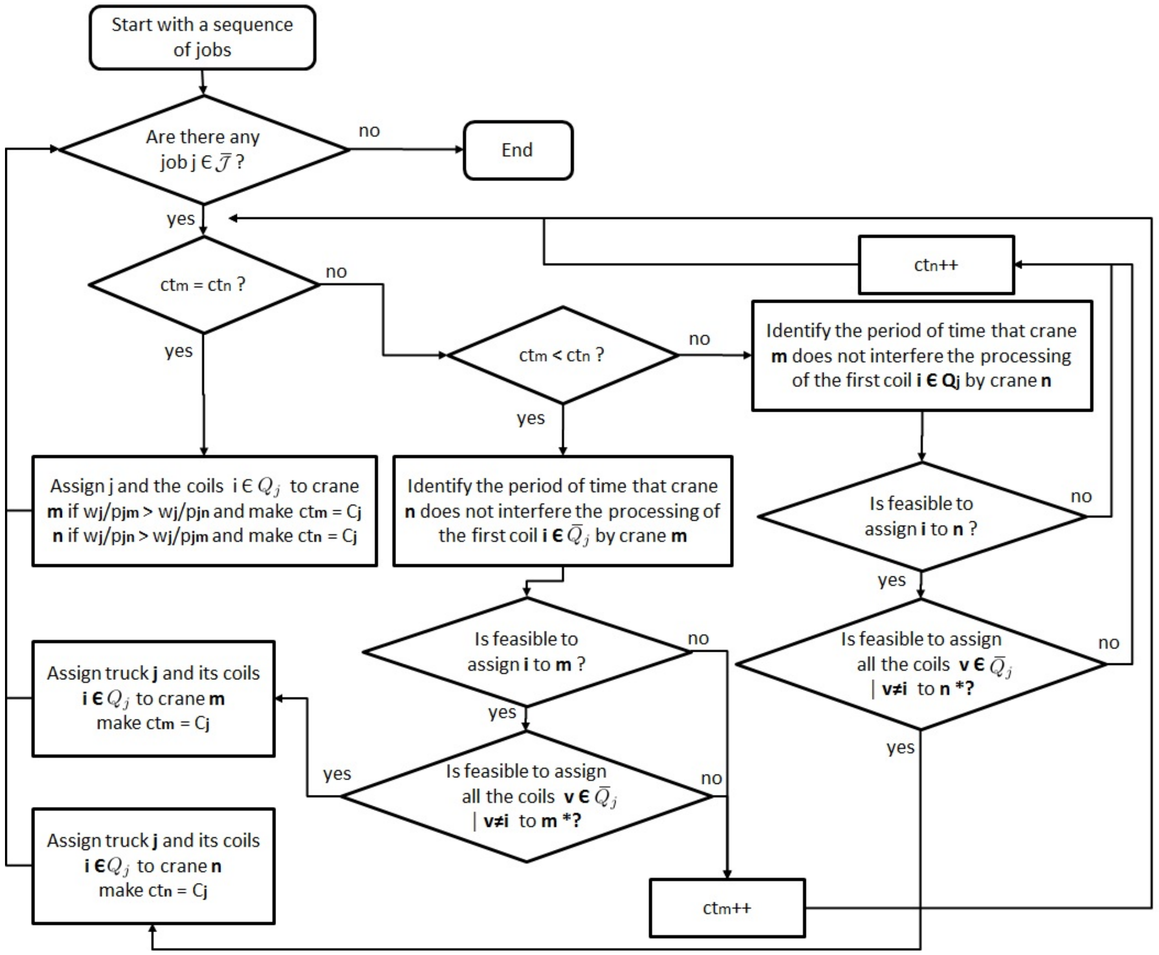
Algorithm 3 Genetic algorithm for the MP2 configuration

```
1: Randomly generate a population of $\mu$ chromosomes of $J$ trucks, each one with $Q_j$ coils.
   Each chromosome represents a candidate solution for the trucks sequencing which have an
   associated sequence of their demanded coils.
2: Calculate the fitness $f(\omega)$ of each chromosome $\omega$ of the population.
3: Create $\lambda$ offsprings.
     • Select a pair of parents chromosomes.
     • With probability $pc$, an order crossover is applied to generate two offspring.
           With probability $pm$, two randomly chosen coils $i \in Q_j$ change their processing
       position order.
     • Two randomly chosen trucks change their position with probability $pm$.
4: Replace the population.
5: Return to step 2.
```

The fitness is calculated as a simple transformation of the objective function in the same way as in $GA1$. But differently, in $GA2$ the scheduling process of trucks and coils considers the non-interference constraints in function of the coil's storing position in the shed. The weighted completion time is the value obtained after assigning the trucks to cranes following the sequence given by the chromosome. This simple heuristic is defined in the following section.

5.3.1 Assignement heuristic

This heuristic should compute the weighted completion time value obtained from the assignment of jobs to machines, following the sequence given by each chromosome. The assignment procedure considers ct_m and ct_n as the current completion time of machine m and n, both $\in \mathcal{M}$, and $ct_m = ct_n = 0$ at time zero. It considers the set $\bar{\mathcal{J}} \subseteq \mathcal{J}$ of unassigned jobs and the set $Q_j \subseteq \mathcal{B}$ of demanded coils from job j, where $\mathcal{B}$ is the set of items that must be delivered in the period of analysis. Consider the set $\bar{Q}_j \subseteq Q_j$ of non-evaluated items. For example, if we evaluate and decide where coil $i \in \bar{Q}_j$ can be processed without activating the non-interference constraints, then i is removed from set $\bar{Q}_j$. This procedure can be verified on Figure 5.4.



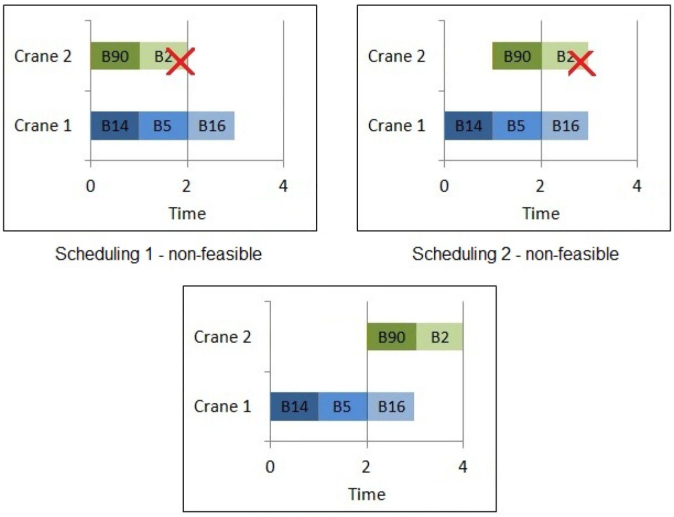
Start with a sequence of jobs
Are there any job j ∈ $\bar{\mathcal{J}}$?
no
End
yes
ctm = ctn ?
no
yes
ctm < ctn ?
no
yes
ctn++
Identify the period of time that crane m does not interfere the processing of the first coil i ∈ Qj by crane n
Assign j and the coils i ∈ Q_j to crane m if wj/pjm > wj/pjn and make ctm = Cj n if wj/pjn > wj/pjm and make ctn = Cj
Identify the period of time that crane n does not interfere the processing of the first coil i ∈ $\bar{Q}_j$ by crane m
Is feasible to assign i to n ?
no
yes
Is feasible to assign i to m ?
no
yes
Is feasible to assign all the coils v ∈ $\bar{Q}_j$ | v≠i to n *?
no
yes
Assign truck j and its coils i ∈ Q_j to crane m make ctm = Cj
Is feasible to assign all the coils v ∈ $\bar{Q}_j$ | v≠i to m *?
yes
no
Assign truck j and its coils i ∈Q_j to crane n make ctn = Cj
ctm++

all the items $i \in Q_j$ must be processed without preemption

Figure 5.4: Flowchart of the proposed sequencing heuristic for the GA2

Returning to the example given on Table 2.1, consider that job $C2$ has coils sequence given by $\{B14,\ B5,\ B16\}$ and $C3$ has items sequence equals to $\{B90,\ B2\}$. Now consider that $C2$ is already assigned to machine 1 and that we need to choose the period to start processing

$C3$ by machine 2. Remember that the number after B is the row of the coil and assume $p_{im} = 1, \forall i \in \mathcal{B}, \forall m \in \mathcal{M}$. Then, $C3$ starts only when the processing of $B2$ and $B90$ on crane 2 is not blocked by crane 1, while it processes truck $C2$, as exemplified on Figure 5.5.



Scheduling 1 - non-feasible

Scheduling 2 - non-feasible

Scheduling 3 - feasible

Figure 5.5: Example of sequencing trucks and coils

5.4 Results of the genetic algorithms

The genetic algorithm heuristics are implemented in C language and the computer used in the tests is an Intel (R) Xeon (R) CPU X5690 @ 3.47GHz with 24 processors, 132 GB of RAM in Ubuntu Linux operating system. The heuristics are tested in the same artificial instances used in $MP1$ and $MP2$ problems. All of the presented results are the average of 10 replications. It is also considered the following coils arrangement scenarios: random scenario (R); scenario of clients (C); customers group scenario (GC), in which R represents the random arrangement of coils in the storage shed; C represents the coils arrangement by customers' row(s) and on GC scenario the coils are arranged by customer groups.

Intending to enable the performance evaluation of $GA1$ and $GA2$, the lower bound is considered as the value of the objective function of a parallel machine problem, without non-interference constraints (PM). However, for the $GA1$'s evaluation, we consider that there is, in PM, a set $\Omega \subseteq \mathcal{J}$ of jobs that must be processed alone, because their loading prevents the processing of all the other jobs. In this way, the optimal solution value of this problem (S^R) is greater than or equals the optimal solution value obtained by the parallel machine problem without interference

(S^{PM}). Considering S^* as the optimal solution obtained by the $MP1$ problem, the relation can be summarized as $S^{PM} \leq S^R \leq S^*$.

Table 5.3 and 5.4 are, respectively, a compilation of $GA1$ and $GA2$ performance on scenarios R, C and GC. The first column of the table represents the instance groups, which are coded by the availability of trucks: nr, sr and vr; and by the amount of steel coils to be delivered: $\{10, 20, 50, 100, 200\}$. The $\sum S^*$ column is the amount of instances optimally solved by the mathematical model; $\sum nb$ is the amount of instances without feasible solution found after one hour of the model's run. GAxS^* column presents the average gap between the optimal solutions found by the mathematical model and the solutions value found by the GAs for the same instances; GAxS^f represents the average gap between the feasible non-optimal solutions found by the model after one hour of CPU time and the GAs' solutions of the same instances.

Column GAxS^{R*} represents the average gap between the lower bound solutions obtained for the same optimally solved instances and the GAs solutions of the same instances; GAxS^{Rf} represents the average gap between the lower bound solutions obtained for the same instances non-optimally solved with the mathematical model and the solution found by the GAs; and GAxLB represents the average gap between the lower bound solutions value and the solutions found by the GAs. CT^* represents the average computation time spent by the mathematical model for the optimal solved cases; and CT^{ga} represents the average computation time spent by the GAs.

The analysis of the results show that $GA1$ and $GA2$ present a high gap when related with the lower bounds from the R scenario. This is due to the high amount of active non-interference constraints caused by the arrangement of coils on this scenario, and the considered lower bounds might present a high gap solution in relation to the real problem optimal solution. This can be seen on the evaluation of the solutions of small size instances, obtained from the comparison of column GAxS^* and GAxS^{R*}. The $GA1$ presents an optimality gap lower than 1.9% when compared with optimal solutions, but has optimality gap up to 10% when compared with the lower bounds of the same group of instances.

However, the genetic algorithms solutions are better than the feasible non-optimal solutions found by the mathematical models. This can be seen through the negative values given on the GAxS^f column.

$GA1$ presents better results than $GA2$, their results are an average of 4% better than the ones of $GA2$. It is easy to see that both GAs's computation time increases with the size of the instance, this is because the amount of generations is defined as a function of the amount of coils. $GA2$ is slower than $GA1$ because crossover and mutation operations are performed to the coils sequences as well. If we limit the running time in 10 minutes, the gaps of the 200 coils' instances will only increase an average of 0.5%. Tables 5.5 and 5.6 present the iteration and the time in which occurred the last improvement o each group of instances.

Table 5.3: Results of the $GA1$ by scenario of coil storage

			GAP						
inst.	$\sum S^*$	$\sum nb$	GAxS^*	GAxS^f	GAxS^{R*}	GAxS^{Rf}	GAxLB	CT^*	CT^{ga}
Random Scenario (Scenario R)									
mpr_10	10	0	0.0%		7.7%		7.7%	1	0
pr_10	10	0	0.0%		5.5%		5.5%	1	0
mr_10	10	0	0.0%		6.1%		6.1%	1	0
mpr_20	10	0	0.0%		10.0%		10.0%	11	0
pr_20	10	0	0.6%		6.8%		6.8%	14	0
mr_20	10	0	0.8%		7.5%		7.5%	14	0
mpr_50	6	0	0.8%	0.4%	6.0%	4.7%	10.8%	1417	4
pr_50	7	0	1.9%	0.3%	7.5%	6.2%	13.7%	1611	5
mr_50	7	0	0.9%	0.1%	7.0%	3.7%	10.7%	1076	5
mpr_100	0	0		-116.8%		11.3%	11.3%		47
pr_100	0	0		-103.7%		13.3%	13.3%		48
mr_100	0	0		-69.9%		12.3%	12.3%		47
mpr_200	0	6		-947.5%		13.7%	13.7%		565
pr_200	0	10					13.4%		508
mr_200	0	10					15.1%		510
Scenario of customers rows (Scenario C)									
mpr_10	10	0	0.0%		5.3%		5.3%	1	0
pr_10	10	0	0.0%		1.0%		1.0%	1	0
mr_10	10	0	0.0%		1.8%		1.8%	1	0
mpr_20	10	0	0.0%		3.5%		3.5%	7	0
pr_20	10	0	0.0%		2.4%		2.4%	9	0
mr_20	10	0	0.0%		1.9%		1.9%	10	0
mpr_50	10	0	0.4%		3.0%		3.0%	196	4
pr_50	10	0	0.9%		1.6%		1.6%	267	5
mr_50	10	0	0.4%		2.9%		2.9%	249	5
mpr_100	3	0	0.2%	-0.5%	0.1%	1.4%	1.6%	2201	47
pr_100	3	0	0.1%	-0.7%	0.1%	1.0%	1.1%	2015	46
mr_100	4	0	0.3%	-0.5%	0.3%	1.0%	1.3%	1618	45
mpr_200	0	0		-1075.2%		1.2%	1.2%		511
pr_200	0	5		-1157.9%		1.3%	1.3%		507
mr_200	0	8		-1224.9%		1.6%	1.6%		523
Scenario of grouping customers rows (Scenario GC)									
mpr_10	10	0	0.0%		4.3%		4.3%	1	0
pr_10	10	0	0.0%		1.2%		1.2%	1	0
mr_10	10	0	0.1%		0.5%		0.5%	1	0
mpr_20	10	0	0.0%		4.2%		4.2%	8	0
pr_20	10	0	0.0%		2.3%		2.3%	8	0
mr_20	10	0	0.2%		2.8%		2.8%	11	0
mpr_50	10	0	0.1%		2.8%		2.8%	345	5
pr_50	10	0	0.9%		1.5%		1.5%	311	5
mr_50	10	0	0.6%		3.1%		3.1%	261	5
mpr_100	3	0	0.3%	-0.2%	0.2%	1.2%	1.4%	2613	50
pr_100	4	0	0.2%	-0.7%	0.2%	1.0%	1.2%	2416	49
mr_100	4	0	0.2%	-0.4%	0.2%	1.3%	1.5%	2000	48
mpr_200	0	0		-1201.9%		1.2%	1.2%		538
pr_200	0	6		-1161.3%		1.4%	1.4%		533
mr_200	0	8		-1215.9%		1.6%	1.6%		534

Table 5.4: Results of the $GA2$ by scenario of coil storage

			GAP						
inst.	$\sum S^*$	$\sum nb$	GAxS^*	GAxS^f	GAxS^{R*}	GAxS^{Rf}	GAxLB	CT^*	CT^{ga}
Random Scenario (Scenario R)									
mpr_10	10	0	0.0%		2.7%		2.7%	21	0
pr_10	10	0	0.1%		4.4%		4.4%	16	0
mr_10	10	0	0.5%		1.8%		1.8%	10	0
mpr_20	0	6		-138.4%		5.6%	5.6%		1
pr_20	4	5	0.2%	0.1%	0.8%	2.4%	3.2%	1030	1
mr_20	4	1	0.8%	-0.9%	0.7%	2.5%	3.2%	1236	1
mpr_50	0	10					5.3%		24
pr_50	0	10					4.2%		26
mr_50	0	10					3.3%		25
mpr_100	0	10					3.5%		242
pr_100	0	10					4.5%		315
mr_100	0	10					4.3%		233
mpr_200	0	10					4.3%		2580
pr_200	0	10					5.0%		2701
mr_200	0	10					5.0%		2704
Scenario of customers rows (Scenario C)									
mpr_10	10	0	0.0%		1.2%		1.2%	13	0
pr_10	10	0	0.0%		0.7%		0.7%	12	0
mr_10	10	0	0.0%		1.7%		1.7%	11	0
mpr_20	2	2	0.0%	-45.3%	0.0%	0.8%	0.8%	1342	1
pr_20	5	2	0.5%	0.1%	0.8%	0.3%	1.1%	749	2
mr_20	7	0	0.0%	-13.7%	0.3%	0.3%	0.6%	781	2
mpr_50	0	10					0.7%		26
pr_50	0	10					0.4%		28
mr_50	0	10					0.7%		27
mpr_100	0	10					0.5%		253
pr_100	0	10					0.4%		251
mr_100	0	10					0.4%		249
mpr_200	0	10					0.4%		2768
pr_200	0	10					0.5%		2721
mr_200	0	10					0.6%		2744
Scenario of grouping customers rows (Scenario GC)									
mpr_10	10	0	0.0%		1.2%		1.2%	15	0
pr_10	10	0	0.3%		1.2%		1.2%	13	0
mr_10	10	0	0.0%		1.0%		1.0%	10	0
mpr_20	1	2	0.0%	-18.5%	0.0%	1.2%	1.2%	1307	1
pr_20	7	1	0.1%	-33.1%	0.4%	0.1%	0.6%	1049	2
mr_20	5	1	0.0%	-0.3%	0.2%	0.1%	0.3%	1178	2
mpr_50	0	10					0.6%		25
pr_50	0	10					0.5%		28
mr_50	0	10					1.1%		27
mpr_100	0	10					0.6%		259
pr_100	0	10					0.3%		253
mr_100	0	10					0.5%		252
mpr_200	0	10					0.6%		2670
pr_200	0	10					0.6%		2594
mr_200	0	10					0.7%		2473

Table 5.5: Data of the last improved solution for the $GA1$

	Generation of the last improve			Time of the last improve		
instances	Minimum	Average	Maximum	Minimum	Average	Maximum
Random Scenario (Scenario R)						
mpr_10	1	2	7	0	0	0
pr_10	1	31	284	0	0	0
mr_10	1	10	52	0	0	0
mpr_20	4	74	485	0	0	0
pr_20	8	62	476	0	0	0
mr_20	4	13	19	0	0	0
mpr_50	25	122	466	0	0	2
pr_50	44	324	736	0	1	3
mr_50	38	152	434	0	1	1
mpr_100	183	899	1660	3	14	27
pr_100	155	1083	2375	2	17	39
mr_100	97	1199	1917	1	18	30
mpr_200	3391	4567	5954	267	431	829
pr_200	2341	4429	5681	182	367	495
mr_200	1543	4004	5488	121	332	461
Scenario of customers rows (Scenario C)						
mpr_10	1	3	6	0	0	0
pr_10	1	3	7	0	0	0
mr_10	1	5	12	0	0	0
mpr_20	6	10	18	0	0	0
pr_20	5	32	166	0	0	0
mr_20	11	106	494	0	0	1
mpr_50	22	291	1088	0	1	2
pr_50	51	373	1469	0	1	5
mr_50	44	264	1008	0	1	4
mpr_100	323	1791	2702	5	28	43
pr_100	405	1610	3071	6	24	44
mr_100	179	1594	2998	2	24	45
mpr_200	3068	4811	5887	263	396	480
pr_200	3392	5073	5995	304	413	475
mr_200	2731	4294	5240	224	363	421
Scenario of grouping customers rows (Scenario GC)						
mpr_10	1	2	5	0	0	0
pr_10	1	2	4	0	0	0
mr_10	1	28	241	0	0	0
mpr_20	4	14	63	0	0	0
pr_20	6	18	70	0	0	0
mr_20	8	16	25	0	0	0
mpr_50	24	123	585	0	1	2
pr_50	47	502	1508	0	2	5
mr_50	35	216	1472	0	1	6
mpr_100	147	1381	2725	3	22	43
pr_100	377	1901	2731	6	31	51
mr_100	391	1831	2978	7	28	45
mpr_200	1908	4314	6161	172	375	536
pr_200	2402	4685	6157	193	406	545
mr_200	2359	4367	6076	194	377	519

Table 5.6: Data of the last improved solution for the $GA2$

	Generation of the last improve			Time of the last improve		
instances	Minimum	Average	Maximum	Minimum	Average	Maximum
Random Scenario (Scenario R)						
mpr_10	1	3	9	0	0	0
pr_10	1	4	9	0	0	0
mr_10	1	8	21	0	0	0
mpr_20	9	131	655	0	0	1
pr_20	12	48	197	0	0	1
mr_20	9	52	203	0	0	0
mpr_50	42	468	2017	0	5	20
pr_50	42	538	2219	0	6	23
mr_50	43	269	743	0	3	9
mpr_100	754	1479	2552	39	79	136
pr_100	388	1817	4007	20	171	950
mr_100	263	2085	4489	14	106	225
mpr_200	1807	5213	8585	479	1469	2387
pr_200	3063	5776	8673	869	1698	2560
mr_200	5928	7692	9138	1758	2262	2775
Scenario of customers rows (Scenario C)						
mpr_10	1	3	6	0	0	0
pr_10	1	3	5	0	0	0
mr_10	1	4	8	0	0	0
mpr_20	1	27	195	0	0	1
pr_20	9	24	125	0	0	1
mr_20	8	17	45	0	0	0
mpr_50	25	155	464	0	2	6
pr_50	34	332	1119	0	4	14
mr_50	30	144	423	0	2	5
mpr_100	357	1950	4581	19	109	253
pr_100	387	870	1758	21	48	93
mr_100	432	1283	4073	24	70	217
mpr_200	3253	7364	9166	993	2224	2866
pr_200	2102	5694	9154	631	1697	2675
mr_200	1661	6036	9194	497	1817	2923
Scenario of grouping customers rows (Scenario GC)						
mpr_10	1	2	6	0	0	0
pr_10	1	3	6	0	0	0
mr_10	3	9	51	0	0	0
mpr_20	5	38	298	0	0	1
pr_20	7	13	24	0	0	1
mr_20	8	14	22	0	0	1
mpr_50	31	143	773	0	2	8
pr_50	38	269	808	0	4	11
mr_50	34	209	700	0	3	8
mpr_100	394	1080	2166	22	61	121
pr_100	194	1675	4215	11	93	236
mr_100	96	1250	3818	6	69	205
mpr_200	5116	7307	9157	1530	2130	2758
pr_200	2114	7077	9055	587	2011	2721
mr_200	3150	6651	8787	777	1813	2537

Chapter 6

Conclusions and further works

This work focuses on the two cranes scheduling problem of a steelmaker's distribution center, while considers the different coils arrangement policies in the storage shed. The cranes must load the trucks with their demanded coils and, on this process, one crane can not overtake the other because both move on the same trail. This crane's displacement relation is treated through non-interference constraints, which often appears at logistic centers and are especially found at port terminals.

In this way, the study analyzes the scenario by different modeling perspectives, which results in different problems with different solutions and computational complexities. We propose two machine configuration paradigms to represent the scenario: parallel machine problems and multiprocessors problem. The scheduling problem can be understood as a multiprocessor problem, if each truck is processed by two machines simultaneously. It also can be considered as parallel machine problem, if each truck is completely loaded by only one of the cranes.

The multiprocessors paradigm addresses one problem referred as $MULTI$, and the parallel machine paradigm addresses two problems: $MP1$ and $MP2$. $MP1$ considers only the trucks scheduling and disregards the retrieval sequence of each coil from a truck. Because of that, it bounds an area in the storage shed for the processed truck and prohibits the other crane to process jobs that have coils in this area. $MP1$ problem is formulated with time index variables and with precedence index variables, but the computational experiments pointed out the time index formulation as the one with best performance. In another way, $MP2$ and $MULTI$ consider the scheduling of trucks and each one of their coils, which leads the non-interference constraints to be function of the coils' position into the shed.

The problems are mathematically modeled and implemented in order to concisely represent and optimally solve the scheduling problems. The optimization criterion is defined as the minimization of the sum of the weighted completion time of jobs, as in the real case, some customers may have processing priorities. All the formulations are implemented in AMPL language, using the routines of the CPLEX 12.5 in standard configuration and the developed tests from da Silva Neto (2013) results.

The $MP1$ configuration presents the best computational performance, since it is able to find optimal solutions for all the instances of up to 20 coils. It can find optimal solutions for 78% of the instances up to 100 coils, and it is able to find non-optimal feasible solutions for larger instances than the other models. $MP2$ finds an average of 2.9% better optimal solutions values than $MP1$ for small instances. $MULTI$ problem is a $MP2$ more complex version and can only find optimal solutions for 10 coils instances. Coils arrangement by rows of customers groups (GC scenario) generates better results for all the models, while grouping coils by customers' rows (C scenario) generates the worst solutions for $MULTI$.

Even if the mathematical models can not optimally solve real size instances, they are important in the problem's characterization and in the evaluation process of heuristic methods. Two genetic algorithms are developed based on the work of Lee et al. (2008) to heuristically solve the parallel machine problems. In order to increase their performance, five genetic algorithm's parameters are tuned with the SPOT method. Both algorithms are implemented in C language and tested in the same artificial instances developed for testing the mathematical models.

The two genetic algorithms present worst solutions for scenario of random coils arrangement (R scenario). This might be because the R scenario is characterized by many active non-interference constraints, and the considered lower bound might present high gap in relation to the optimal solution. This does not necessarily indicates that the genetic algorithms did not find near optimal or optimal solutions.

The genetic algorithm developed to solve the $MP1$ problem presents results with an average of 10.5% gap from the lower bound in the R scenario, and with an average of 2.1% gap in the C

and GC scenarios. While the genetic algorithm for solving the $MP2$ problem presents results with an average of 4% gap from the lower bound in the R scenario, and 0.7% gap in the in the C and GC scenarios. These high gaps resulted from the R scenario are due to the high amount of active non-interference constraints caused by the arrangement of coils.

There are several possible research directions to expand the work made in this dissertation, we can mention, novel reformulation schemes for modeling the non-interference constraints, new algorithms and approaches to efficiently solve real size instances, consideration of more real characteristics, such as stochastic processing times, breakdowns, etc and the integration of the scheduling decisions in the company's Supply Chain Management.

Bibliography

Afrati, F., Bampis, E., Chekuri, C., Karger, D., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., et al. (1999). Approximation schemes for minimizing average weighted completion time with release dates. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 32–43. IEEE.

Allahverdi, A., Gupta, J. N. D., and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032.

Bäck, T. (1992). The interaction of mutation rate, selection and self-adaptation within a genetic algorithm. *Parallel Problem Solving from Nature 2*, pages 85–94.

Bäck, T. and Schütz, M. (1996). Intelligent mutation rate control in canonical genetic algorithms. In *Foundations of intelligent systems*, pages 158–167. Springer.

Baker, K. R. (1974). *Introduction to sequencing and scheduling*, volume 31. Wiley New York.

Bartz-Beielstein, T. (2010). Spot: An r package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. *arXiv preprint arXiv:1006.4645*.

Bartz-Beielstein, T. (2014). Package spot. *Reference manual, `http://cran.r-project.org/web/packages/SPOT/index.html`. Cited 11 January 2015.*

Bartz-Beielstein, T. and Zaefferer, M. (2012). A gentle introduction to sequential parameter optimization. Technical Report 2, Bibliothek der Fachhochschule Koeln.

Belouadah, H., Posner, M. E., and Potts, C. N. (1992). Scheduling with release dates on a single machine to minimize total weighted completion time. *Discrete Applied Mathematics*, 36(3):213–231.

Belouadah, H. and Potts, C. N. (1994). Scheduling identical parallel machines to minimize total weighted completion time. *Discrete Applied Mathematics*, 48(3):201–218.

Bierwirth, C. and Meisel, F. (2009). A fast heuristic for quay crane scheduling with interference constraints. *Journal of Scheduling*, 12(4):345–360.

Bierwirth, C. and Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202(3):615–627.

Chen, J. and Lee, C.-Y. (1999). General multiprocessor task scheduling. *Naval Research Logistics*, 46(1):57–74.

Cheng, T. C. E. and Sin, C. C. S. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47(3):271–292.

Chung, S. H. and Choy, K. L. (2012). A modified genetic algorithm for quay crane scheduling operations. *Expert Systems with Applications*, 39:4213—-4221.

Conway, R. W., Maxwell, W. L., and Miller, L. W. (1967). *Theory of scheduling*. DoverPublications. com.

da Silva Neto, J. P. (2013). Montagem de cargas e sequenciamento de caminhões em um centro de distribuição. *Universidade Federal de Minas Gerais, Brasil. Dissertação de Mestrado.*

de Paula, M. R., Mateus, G. R., and Ravetti, M. G. (2010). A non-delayed relax-and-cut algorithm for scheduling problems with parallel machines, due dates and sequence-dependent setup times. *Computers & Operations Research*, 37(5):938–949.

Drozdowski, M. (1996). Scheduling multiprocessor tasks—an overview. *European Journal of Operational Research*, 94(2):215–230.

Eiben, A. and Smit, S. (2012). Evolutionary algorithm parameters and methods to tune them. In *Autonomous Search*, pages 15–36. Springer.

Eiben, A. E., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 3(2):124–141.

Eiben, A. E., Michalewicz, Z., Schoenauer, M., and Smith, J. E. (2007). Parameter control in evolutionary algorithms. In *Parameter setting in evolutionary algorithms*, pages 19–46. Springer.

Eiben, A. E. and Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31.

Gaiu, A. F. (2013). Sequential parameter tuning of algorithms for the vehicle routing problem. *Leiden Institute of Advanced Computer Science, Netherlands. Master's Thesis.*

Gen, M. and Cheng, R. (2000). *Genetic algorithms and engineering optimization*, volume 7. John Wiley & Sons.

Glass, C., Potts, C., and Shade, P. (1994). Unrelated parallel machine scheduling using local search. *Mathematical and Computer Modelling*, 20(2):41–52.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics. v5*, pages 287–326.

Gupta, J. N. D. and Stafford Jr, E. F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699–711.

Hall, N. G. and Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations research*, 44(3):510–525.

Herrmann, J. W. (2006). A history of production scheduling. In *Handbook of Production Scheduling*, pages 1–22. Springer.

Hoogeveen, J. A., van de Velde, S. L., and Veltman, B. (1994). Complexity of scheduling multiprocessor tasks with prespecified processor allocations. *Discrete Applied Mathematics*, 55(3):259–272.

Hoos, H. H. and Stützle, T. (2004). *Stochastic local search: Foundations & applications.* Elsevier.

IAB (2013). Instituto aço brasil, relatório de sustentabilidade. *Disponível em: http://www.acobrasil.org.br. Acessado em 22 de janeiro de 2014.*

Kim, K. H. and Park, Y.-M. (2004). A crane scheduling method for port container terminals. *European Journal of operational research*, 156(3):752–768.

Kravchenko, S. A. and Werner, F. (2011). Parallel machine problems with equal processing times: a survey. *Journal of Scheduling*, 14(5):435–444.

Lee, C.-Y. and Cai, X. (1999). Scheduling one and two-processor tasks on two parallel processors. *IIE transactions*, 31(5):445–455.

Lee, C.-Y., Lei, L., and Pinedo, M. (1997). Current trends in deterministic scheduling. *Annals of Operations Research*, 70:1–41.

Lee, D.-H., Wang, H. Q., and Miao, L. (2008). Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 44(1):124–135.

Lenstra, J. K., Kan, A. H. G. R., and Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362.

Li, K. and Yang, S.-l. (2009). Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Applied mathematical modelling*, 33(4):2145–2158.

Li, W., Wu, Y., Petering, M. E., Goh, M., and Souza, R. d. (2009). Discrete time model and algorithms for container yard crane scheduling. *European Journal of Operational Research*, 198(1):165–172.

Lim, A., Rodrigues, B., and Xu, Z. (2007). A m-parallel crane scheduling problem with a non-crossing constraint. *Naval Research Logistics (NRL)*, 54(2):115–127.

Lis, J. and Lis, M. (1996). Self-adapting parallel genetic algorithm with the dynamic mutation probability, crossover rate and population size. In *Proceedings of the 1st Polish National Conference on Evolutionary Computation*, pages 324–329. Oficina Wydawnica Politechniki Warszawskiej.

Maschietto, G., Ouazene, Y., Yalaoui, F., de Souza, M., and Ravetti, M. (2015). Two formulations for non-interference parallel machine scheduling problems. *15th Symposium on Information Control Problems in Manufacturing, Ottawa, Canada.*

Maschietto, G., Ravetti, M., and de Souza, M. (2014). Two approaches of scheduling problems in a distribution center with two cranes and interference constraint. *20th Conference of The International Federation of Operational Research Societies, Barcelona, Spain.*

McNaughton, R. (1959). Scheduling with deadlines and loss functions. *Management Science*, 6(1):1–12.

Michalewicz, Z. (1995). Genetic algorithms, numerical optimization, and constraints. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, volume 195, pages 151–158. Morgan Kaufmann, San Mateo, CA.

Michalewicz, Z. (1996). *Genetic algorithms+ data structures= evolution programs*. springer.

Min, L. and Cheng, W. (1999). A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. *Artificial Intelligence in Engineering*, 13(4):399–403.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Nessah, R., Chu, C., and Yalaoui, F. (2007). An exact method for problem. *Computers & Operations Research*, 34(9):2840–2848.

Ng, W. (2005). Crane scheduling in container yards with inter-crane interference. *European Journal of Operational Research*, 164(1):64–78.

Pfund, M., Fowler, J. W., and Gupta, J. N. (2004). A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems. *Journal of the Chinese Institute of Industrial Engineers*, 21(3):230–241.

Pinedo, M. (2008). *Scheduling: theory, algorithms, and systems*. Springer.

Rocha, P. L., Ravetti, M. G., Mateus, G. R., and Pardalos, P. M. (2008). Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Computers & Operations Research*, 35(4):1250–1264.

Shim, S.-O. and Kim, Y.-D. (2007). Scheduling on parallel identical machines to minimize total tardiness. *European Journal of Operational Research*, 177(1):135–146.

Skutella, M. and Woeginger, G. J. (1999). A ptas for minimizing the weighted sum of job completion times on parallel machines. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 400–407. ACM.

Tang, L., Xie, X., and Liu, J. (2014). Crane scheduling in a warehouse storing steel coils. *IIE Transactions*, 46(3):267–282.

Thierens, D. (2002). Adaptive mutation rate control schemes in genetic algorithms. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, pages 980–985. IEEE.

Xie, X., Zheng, Y., and Li, Y. (2014). Multi-crane scheduling in steel coil warehouse. *Expert Systems with Applications*, 41(6):2874–2885.

Yalaoui, F. and Chu, C. (2002). Parallel machine scheduling to minimize total tardiness. *International Journal of Production Economics*, 76(3):265–279.

Zäpfel, G. and Wasner, M. (2006). Warehouse sequencing in the steel supply chain as a generalized job shop model. *International Journal of Production Economics*, 104(2):482–501.

Zhu, Y. and Lim, A. (2006). Crane scheduling with non-crossing constraint. *Journal of the Operational Research Society*, 57(12):1464–1471.

Appendix A

Sequential Parameter Optimization

Due to its high efficiency, the Sequential Parameter Optimization Toolbox (SPOT) is the defined method to determine the parameters vector of the genetic algorithm (GA) to solve the $MP1$ problem ($GA1$) and the $MP2$ problem ($GA2$), described on Chapter 5. A parameters vector comprises the values of different parameters of the GA, i.e. it represents a candidate solution of a parameter setting. SPOT is an implementation of the Sequential Parameter Optimization (SPO), which is an iterative model-based method of tuning algorithms. The tuning process is based on the parameters data, and their utility delivered by the performance of the genetic algorithm. SPO performs a multi-stage procedure where the model is updated at each iteration with a set of new vectors and new predictions of utility in order to improve the algorithm's efficiency.

The goal of SPOT is the determination of good parameters settings for heuristic algorithms. It provides statistical tools for analyzing and understanding algorithm's performance. SPOT is implemented as a R package, and is available in the R archive network at `http://cran.r-project.org/web/packages/SPOT/index.html`. Further explanation about SPOT can be obtained from Bartz-Beielstein (2010), which provides an exemplification on how SPOT can be used for automatic and iterative tuning. Bartz-Beielstein and Zaefferer (2012) also give an introductory overview about tuning with SPOT.

The key elements of the SPOT methodology are algorithm design (D_a) and problem design (D_p). The first one defines ranges of parameter values that influence the behaviour of an algorithm, such as crossover rate. These parameters are treated as variables $p_a \in D_a$ in the tuning algorithm, where p_a represents the vector of parameters settings. D_p refers to variables related to the tuning optimization problem, e.g. the search space dimension.

SPOT is composed by two phases: the build of the model and its sequential improvement, as can be see on Algorithm 4. Phase 1 determinates an initial designs population from the algorithm's parameter space. The observed algorithm is run k times for each design, where k is the number of repetitions performed for each parameter setting and is increased in each run. Phase 2 leads to the efficiency of the approach and is characterized by the following loop:

- update the model given the obtained data;
- generate design points and predict their utility by sampling the model;
- choose the best design vectors and run $k + 1$ times the algorithm for each of them;
- new design points are added to the population and the loop restarts if the termination criteria is not reached.

The loop simulates the use of different parameters settings, it is subject to interactions between parameters and random effects into the experiment. SPOT uses results from algorithm runs to build up a meta model to tune algorithms in a reproductible way. The SPOT algorithm is exemplified by Algorithm 4.

A.1 Experimental setup

Classical works about SPOT define three different layers to analyse parameter tuning. The first one is the *application layer*, considered in our case as the parallel machines paradigm with non-interference constraints. The objective function and the problem parameters are defined at this layer. The *algorithm layer*, is related to the representation of the heuristic algorithm and its required parameters are the ones that determine the algorithm's performance. And finally, the *design layer*, where is the tuning method, tries to find good parameter settings for the algorithm layer.

In this way we face two optimization problems: problem solving and parameter tuning. The

Algorithm 4 SPOT pseudocode (Bartz-Beielstein, 2010)

```
Phase 1: building the model
let $A$ be the tuned algorithm;
generate an initial population $\vec{P} = \{\vec{p}^1, \dots, \vec{p}^m\}$ of $m$ parameter vectors;
for each $\vec{p}^i \in \vec{P}$ do
    • run $A$ with $\vec{p}^i$ $k$ times to determine the estimated utility $u^i$ (e.g., average function value
      from 10 runs) of $\vec{p}^i$;
Phase 2: using and improving the model
while termination criterion not true do
    • let $\vec{p}^*$ denote the parameter vector from $\vec{P}$ with best estimated utility;
    • let $k$ the number of repeats already computed for $\vec{p}^*$;
    • build prediction model $F$ based on $\vec{P}$ and $u^1, \dots, u^{|\vec{P}|}$;
    • generate a set $\vec{P}'$ of $l$ new parameter vectors by random sampling;
    • for each $\vec{p}^i \in \vec{P}'$ do
          calculate $f(\vec{p}^i)$ to determine the predicted utility $F(\vec{p}^i)$;
    • select set $\vec{P}$" of $d$ parameter vectors from $\vec{P}'$ with best predicted utility ($d << l$);
    • run $A$ with $\vec{p}^*$ once and recalculate its estimated utility using all $k+1$ test results;
      //(improve confidence)
    • update $k$, e.g., let $k = k+1$;
    • run $A$ $k$ time with each $\vec{p}^i \in \vec{P}$" to determine the estimated utility $F(\vec{p}^i)$; extend the
      population by $\vec{P} = \vec{P} \cup \vec{P}$";
```

problem solving covers the application layer and the GA of the algorithm layer and aims to find an optimal solution for the paradigm. The parameters tuning uses a tuning method to find the best parameter values for the GA. The fitness value is the quality measure of the first optimization, which depends on the problem instance to be solved. Utility is the quality measure for parameter tuning, which reflects the performance of the GA for a vector of parameters.

Thereby, the experimental setup consists of an application layer represented by six instances of the parallel machine paradigm, showed on Table A.1. We randomly select two instances from the group of size 20,100 and 200 coils. The genetic algorithm for the $MP1$ problem ($GA1$) and for the $MP2$ problem ($GA2$) are in the algorithm layer. On the tuning layer is the SPOT method with default SPOT-parameters, which works well for 90% of the users (Bartz-Beielstein, 2014), their default values can be checked on the SPOT manual. To define the region of interest (ROI) of the tuning algorithm, the type and the lower and upper bounds of the GA's parameters are determined as explained on Chapter 5 and are summarized on Table A.2.

Table A.1: sampling of instances

Type	Instances from GA1	Instances from GA2
$Y1$	t3_1_20_7	t2_1_20_8
$Y2$	t3_1_100_2	t1_2_100_1
$Y3$	t1_3_200_1	t2_3_200_6
$Y4$	t1_2_20_2	t1_1_20_10
$Y5$	t1_3_100_9	t1_2_100_5
$Y6$	t2_1_200_8	t3_2_200_5

Table A.2: Parameter bounds for tuning the genetic algorithms

Parameter	Lower bound	Upper bound	Type
Population size (μ)	$B/2$	$10B$	integer
Replacement operator (ro)	$0 = (\mu + \lambda)-$selection	$1 = (\mu, \lambda)-$selection	integer
Crossover probability (pc)	0.6	1	float
Mutation probability (pm)	0.001	0.01	float
Amount of generations (N_{max})	$1B$	$50B$	integer

where B is the size of the instance, given by the amount of coils to be delivered

A.2 Results of the experiment

The SPOT algorithm, implemented in R package, is connected with the genetic algorithms implemented on C with the aid of two callStrings, as presented bellow. The computer used in the tests is an Intel (R) Xeon (R) CPU X5690 @ 3.47GHz with 24 processors, 132 GB of RAM, and Ubuntu Linux operating system.

```
setwd(“/home/Documents/GA_SPOT/”)
callString1 < − paste(“gcc -o comp main.c”)
call1 < − system(callString1, intern=TRUE)

callString2 < − paste(“./comp”, $\mu$, ro, pc, pm, $N_{max}$)
call2 < − system(callString2, intern=TRUE)

# read the results
setwd(“/home/Documents/GA_SPOT/”)
y = read.table(“result.txt”)
```

The results obtained by the six instances in each iteration are normalized, in order to allow their comparison. In the normalization, given by $\bar{Y_i}$, zero represents the lowest value found by instance i in the test, and 1 represents the highest value found by this instance. All the other values are obtained from the relation $\frac{Yip-min_i}{max_i-min_i}$, where Y_{ip} is the solution value of instance i for the set of parameters p. min_i and max_i are, respectively, the minimun and the maximum solution values of instance i found in the tuning process.

We aim to find the best group of parameters, which we assume as the one with the lowest sum of the normalized solution values of the six instances. This might not be the best statistical solution, but it is the best solution indicated by the experiment without any statistical analysis.

Table A.3 and A.4 presents the five best results found by SPOT method for $GA1$ and $GA2$, repectively. The first column indicates the iteration of SPOT, the second one presents the indicated parameters vector. Columns three to eight indicate the solution value obtained by the tested instance, columns nine to fourteen indicate the normalized solution value of the respective instances and finally, column fifteen presents the sum of the normalized results.

The best parameters vector indicated for $GA1$ is population size $(\mu) = 7B/2$, and amount of generations (N_{max}) equals $41B$, where B represents the size of the instance. Replacement operator (ro) is given by $(\mu+\lambda)-$selection, crossover probability (pc) equals 0.8827 and mutation probability (pm) equals 0.0057. For $GA2$ the best vector is given by population size $\mu = 7B/2$, amount of generations $N_{max} = 41B$, replacement operator $ro = (\mu + \lambda)-$selection, crossover probability $pc = 0.8827$, and mutation probability $pm = 0.0057$.

Table A.3: Five best parameters settings for the $GA1$

ite-	parameters set	solution value of the instances						normalized solution value of the instances						
raction	pz/ ro/ pc/ pm/ ng	$Y1$	$Y2$	$Y3$	$Y4$	$Y5$	$Y6$	$\bar{Y}1$	$\bar{Y}2$	$\bar{Y}3$	$\bar{Y}4$	$\bar{Y}5$	$\bar{Y}6$	sum
122	8 / 1 / 0.9705 / 0.0037 / 31	104.84	1774.23	6584.05	152.52	2572.16	7316.65	0	0.0289	0.0467	0	0.0241	0.0042	0.1039
166	10 / 0 / 0.8338 / 0.0043 / 27	104.84	1774.93	6619.30	152.52	2562.19	7333.45	0	0.0335	0.0595	0	0.0049	0.0106	0.1086
162	10 / 0 / 0.8338 / 0.0043 / 27	104.84	1776.85	6559.85	152.52	2559.61	7396.67	0	0.0461	0.0379	0	0.0000	0.0350	0.1190
121	10 / 1 / 0.9724 / 0.0026 / 46	104.84	1775.89	6510.34	152.52	2584.88	7350.35	0	0.0398	0.0199	0	0.0485	0.0171	0.1254
31	10 / 0 / 0.9947 / 0.0035 / 39	104.84	1774.55	6571.76	152.52	2573.73	7394.48	0	0.0310	0.0423	0	0.0271	0.0341	0.1345

Table A.4: Five best parameters settings for the $GA2$

ite-	parameters set	solution value of the instances						normalized solution value of the instances						
raction	pz/ ro/ pc/ pm/ ng	$Y1$	$Y2$	$Y3$	$Y4$	$Y5$	$Y6$	$\bar{Y}1$	$\bar{Y}2$	$\bar{Y}3$	$\bar{Y}4$	$\bar{Y}5$	$\bar{Y}6$	sum
134	10 / 0 / 0.8973 / 0.0064 / 46	86.72	1966.82	5759.18	126.62	2074.00	5751.40	0	0.0141	0.0261	0.1073	0	0.0316	0.1791
128	10 / 0 / 0.8973 / 0.0064 / 46	86.72	1999.19	5733.90	124.88	2097.92	5682.20	0	0.1007	0.0099	0.0292	0.0775	0	0.2174
142	10 / 0 / 0.7590 / 0.0035 / 30	86.72	1961.56	5746.41	126.26	2106.42	5713.57	0	0	0.0179	0.0910	0.1050	0.0143	0.2283
131	10 / 0 / 0.8973 / 0.0064 / 46	86.72	1978.23	5740.61	126.62	2087.30	5751.51	0	0.0446	0.0142	0.1073	0.0431	0.0317	0.2409
178	10 / 0 / 0.8525 / 0.0056 / 37	86.72	1991.54	5742.38	126.62	2082.92	5748.23	0	0.0803	0.0153	0.1073	0.0289	0.0302	0.2619